HAUTE ECOLE VALAISANNE

Travail de diplôme

Développement de nouvelles fonctionnalités pour le logiciel bSol

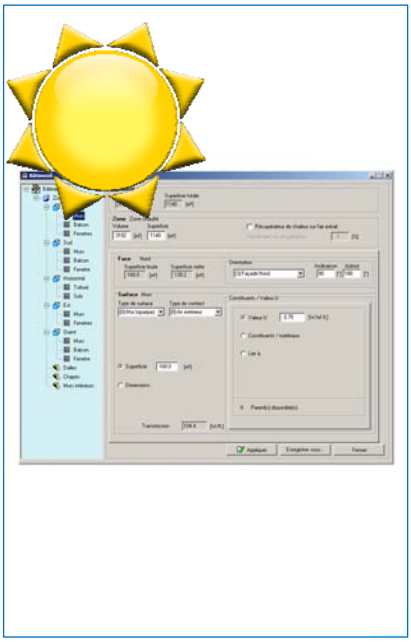
Chin Carvoeiro Sergio

13/11/2009

Améliorations à différents niveaux et programmation de nouvelles fonctionnalités au sein du logiciel bSol.

Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Développement de nouvelles fonctionnalités pour le logiciel bSol



Diplômant Sergio Chin Carvoeiro

Objectif du projet

Le but de ce travail de diplôme est avant tout de comprendre le logiciel bSol et d'en améliorer le fonctionnement sur différents aspects. Ces améliorations englobent des aspects pratiques, des aspects fonctionnels et même cosmétiques, elles vont principalement faciliter l'utilisation du logiciel.

Méthodes | Expériences | Résultats

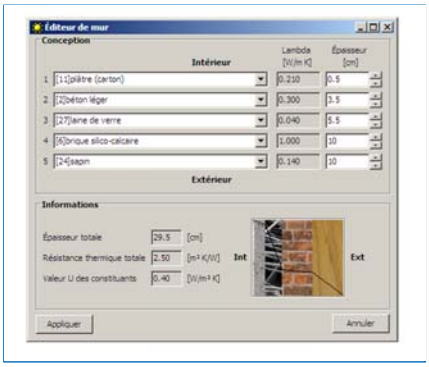
La méthode de travail utilisée ici repose sur le principe du diagramme en forme de V. D'une part, on définit le cahier des charges fonctionnel, les spécifications générales, les spécifications détaillées (conception) et enfin, le codage proprement dit. D'autre part, on exécute les tests unitaires, les tests d'intégration et enfin le programme final.
Le logiciel utilisé pour créer et programmer bSol s'appelle Bolrand© C++ builer. C'est un environnement de développement pour les systèmes d'exploitation Windows©. Le langage employé est le C++, un langage de programmation orientée objet.
En dépit du fait que toutes les fonctionnalités voulues n'ont pas pu être finalisées, les différents tests se sont soldés par un succès. Toutes les fonctionnalités implantées et finalisées au sein de bSol sont utilisables.

Travail de diplôme
| édition 2009 |

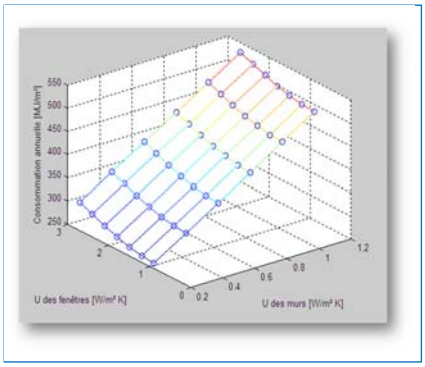
Filière
Systèmes industriels

Domaine d'application
Power and Control

Professeur responsable
Gilbert-André Morand
gandre.morand@hevs.ch



Le nouveau module de construction des murs avec un affichage des différentes couches et de la droite d'échange thermique.



A l'aide de la nouvelle fonction de bSol permettant de générer des fichiers à la volée, il est facile d'étudier la consommation d'un bâtiment.

HES-SO Valais
Route du Rawyl 47
1950 Sion

Tél. 027 606 85 11
URL www.hevs.ch

SOMMAIRE

TRAVAIL DE DIPLÔME

INTRODUCTION

Le logiciel bSol fonctionne depuis 2003 dans sa version actuelle. De nouvelles fonctionnalités, à différents niveaux, devraient maintenant être ajoutées à ce logiciel pour l'adapter à la demande.

Au niveau de l'exploitation d'un bâtiment, l'intégration harmonieuse des consignes horaires de température et des débits de ventilation permettent d'estimer rapidement les économies énergétiques liées à un judicieux choix de ces consignes.

Au niveau du bâtiment lui-même, la mise en place d'un lien entre les mêmes types de matériaux utilisés pour décrire le bâtiment rendront les analyses nécessitant de multiples variations plus aisées.

Une fois l'analyse de la situation faite, et le concept d'intégration défini, il s'agit de programmer dans le logiciel existant, les nouvelles fonctionnalités. Des procédures de tests seront définies et effectuées pour s'assurer que les nouvelles fonctionnalités donnent des résultats de calculs toujours cohérents et qu'elles ne perturbent pas les fonctionnalités déjà existantes.

OBJECTIF

Le but de ce travail de diplôme est avant tout de comprendre le logiciel bSol et d'en améliorer le fonctionnement sur différents aspects. Ces améliorations englobent des aspects pratiques, des aspects fonctionnels et même cosmétiques, elles vont principalement faciliter l'utilisation du logiciel.

NOUVELLES FONCTIONNALITÉS

- Liaison des propriétés (parent-enfant) entre les surfaces de même type.
- Création des murs par l'intermédiaire d'un module différent.
- Définition des consignes de température de manière horaire, journalière et hebdomadaire.

CAHIER DES CHARGES

FONCTIONNALITÉS PRINCIPALES

- Concevoir un module (à la manière du logiciel U-cad) destiné à construire un mur de façon efficace, tout d'abord sous la forme d'exécutable puis comme une application intégrée à bSol. Si possible, disposer d'une représentation graphique des différents matériaux composant le mur ainsi que de l'échange de chaleur entre ces différents matériaux.
  Le même système est à envisager pour les fenêtres et les portes.

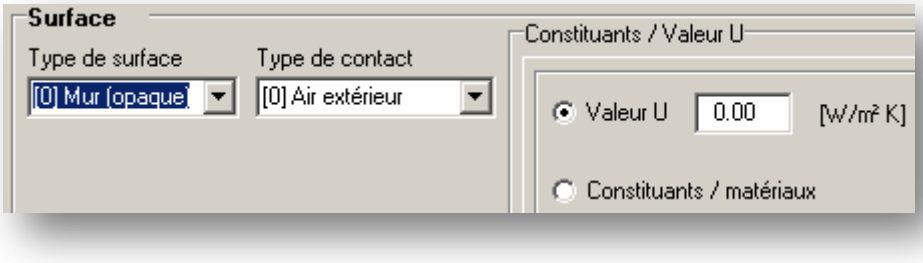


Figure 1: Partie surface

- Permettre de lier plusieurs surfaces de même type à une seule. Ainsi, toute modification apportée à la surface "parent" serait reportée sur toutes les surfaces "enfant". La suite logique de cette fonction étant la conception d'un système permettant de générer facilement plusieurs fichiers "bâtiment" en faisant varier la valeur U des surfaces "parent" afin d'utiliser la fonction *Calcul batch* de bSol.

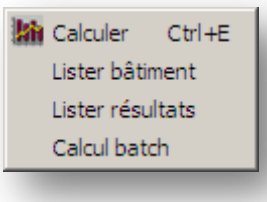


Figure 2: Commande pour le calcul batch

- Dans les parties *Chauffage*, *Aération* et *Climatisation*, on aimerait disposer d'une consigne dynamique qui puisse être réglée en fonction de l'heure, du jour et de la semaine comme celle des *Gains internes*.

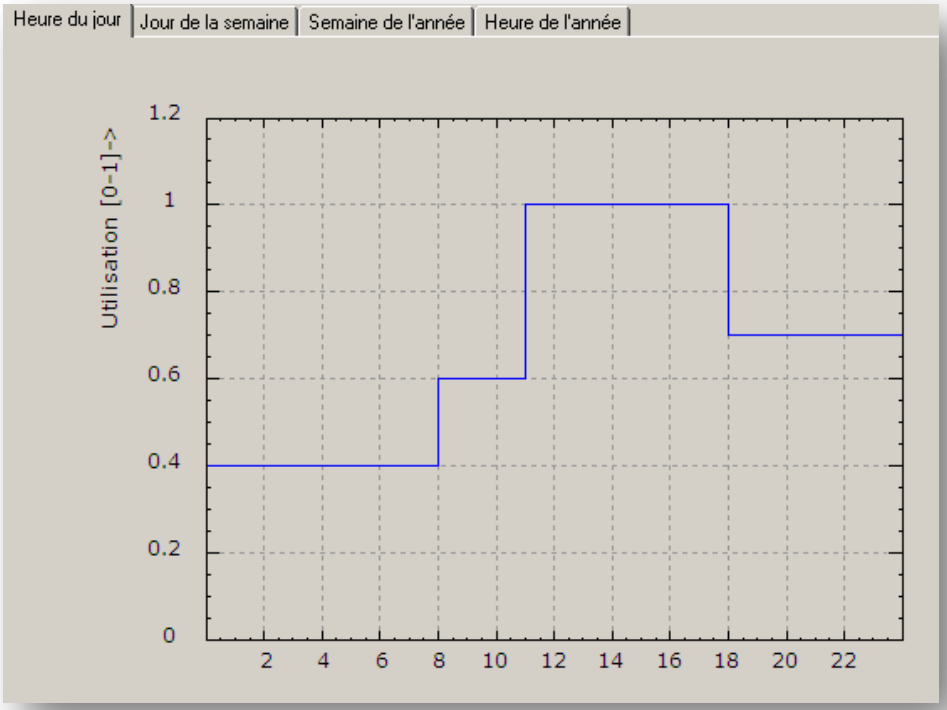


Figure 3: Consigne de température

FONCTIONNALITÉS SECONDAIRES (SI TEMPS DISPONIBLE)

- Envisager un module de calcul pour la profondeur du terrain.
- Améliorer la justification pour la norme SIA 380/1.
- Améliorer la présentation du rapport.
- Actualiser l'interface graphique de bSol.

MÉTHODOLOGIE ET DÉVELOPPEMENT

DIAGRAMME EN V

La méthode de travail utilisée ici repose sur le principe du diagramme en forme de V.

Sur la branche descendante (gauche) du V, on retrouve, de haut en bas, le cahier des charges fonctionnel, les spécifications générales, les spécifications détaillées (conception) et enfin, à la pointe du V, le codage proprement dit.

Sur la branche montante (droite) du V, on trouve les différentes étapes de validation: tests unitaires (pour valider les différents modules de la spécification détaillée), les tests d'intégration (pour valider les spécifications générales) et enfin la recette (pour vérifier que les prescriptions du cahier des charges ont bien été respectées).

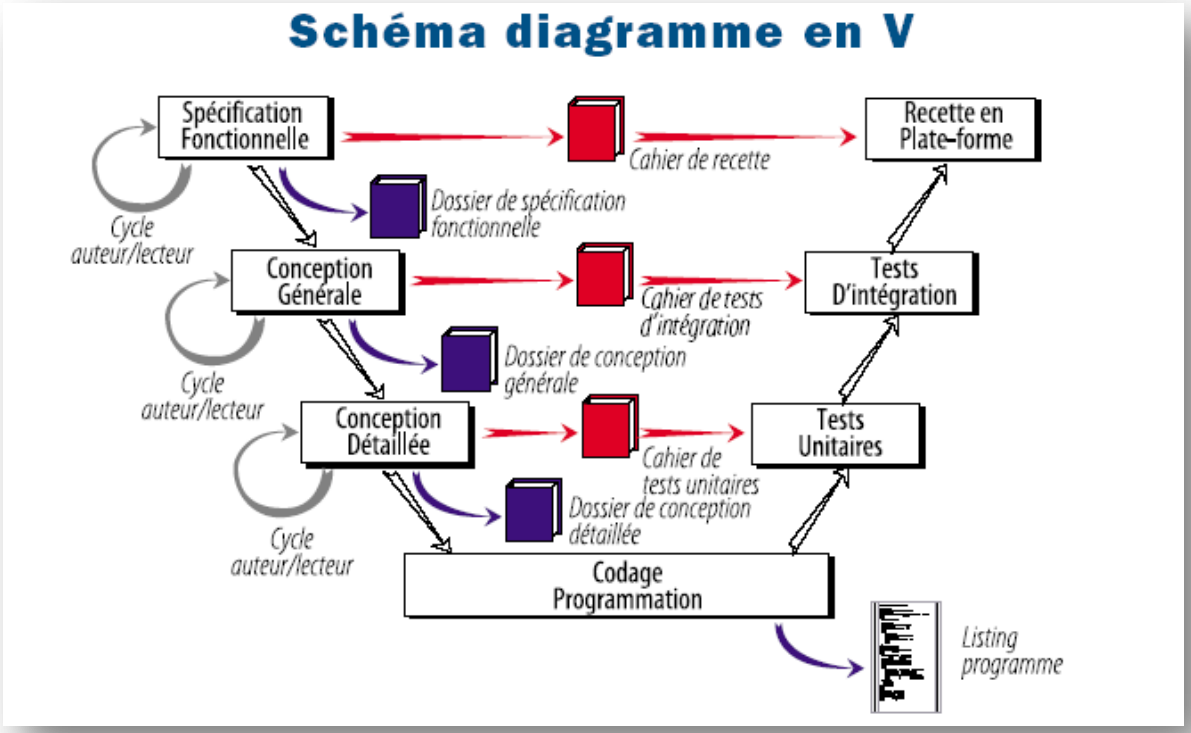


Figure 4: Schéma de diagramme en V

OUTILS DE DÉVELOPPEMENT

Le logiciel utilisé pour créer et programmer bSol s'appelle Bolrand© C++ builer. C'est un environnement de développement pour les systèmes d'exploitation Windows©. Le langage employé est le C++, un langage de programmation orientée objet.

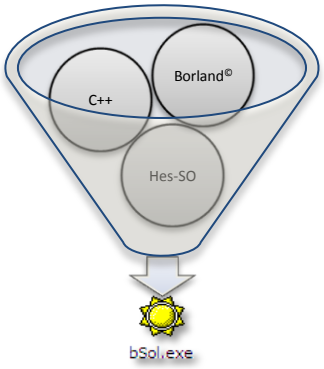


Figure 5: Outils de développement

La programmation orientée objet (POO), est un paradigme de programmation informatique qui consiste en la définition et l'assemblage de briques logicielles appelées *objets*; un objet représente un concept, une idée ou toute entité du monde physique, comme une voiture, une personne ou encore un bâtiment.

CONCEPTS IMPORTANTS EN C/C++

LA VARIABLE

En programmation, une variable est destinée à contenir une valeur du type avec lequel elle est déclarée. Physiquement cette valeur se situe en mémoire.

Exemple: en écrivant **int x;**

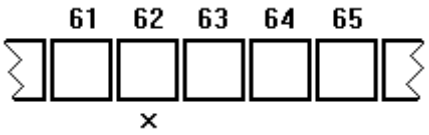


Figure 6: Déclaration d'une variable

on réserve une case mémoire (par ex. la case n° 62) pour la variable x.

En écrivant **x = 10;**

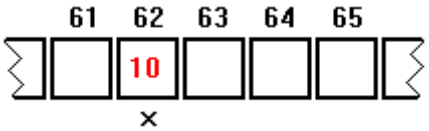


Figure 7: Assignation d'une variable

on écrit la valeur 10 dans l'emplacement réservé pour x. Pour obtenir l'adresse d'une variable, on fait précéder son nom de l'opérateur '&'.

LE POINTEUR

Un pointeur est aussi une variable, il est destiné à contenir une adresse mémoire, c'est à dire une valeur identifiant un emplacement en mémoire. Pour différencier un pointeur d'une variable ordinaire, on fait précéder son nom du signe '*' lors de sa déclaration.

Exemple: en écrivant **int *px;**

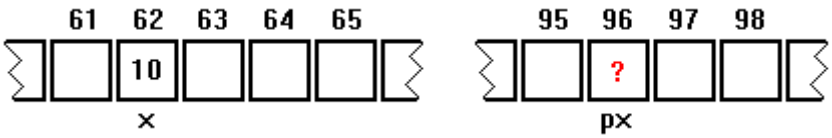


Figure 8: Déclaration d'un pointeur

on réserve un emplacement en mémoire (la case 96) pour le pointeur px.

En écrivant **px = &x;**

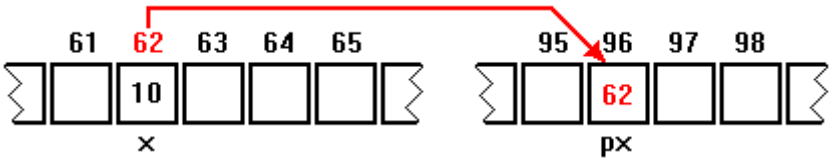


Figure 9: Assignation d'un pointeur

on écrit l'adresse de x à l'emplacement réservé pour le pointeur px.

BSOL

DESCRIPTION

bSol est avant tout un logiciel développé à la Hes-SO. Il permet de déterminer les besoins énergétiques d'une habitation en fonction de ses paramètres architecturaux (composition des murs, surfaces vitrées, etc.), de données météorologiques et d'effets d'horizon. Mais comment parler de bSol sans évoquer la *boussole* (potentiel d'amélioration); il s'agit en fait de la capacité du logiciel à guider l'utilisateur, de façon dynamique et graphique, sur le choix des travaux à effectuer tout en tenant compte d'un facteur de prépondérance.

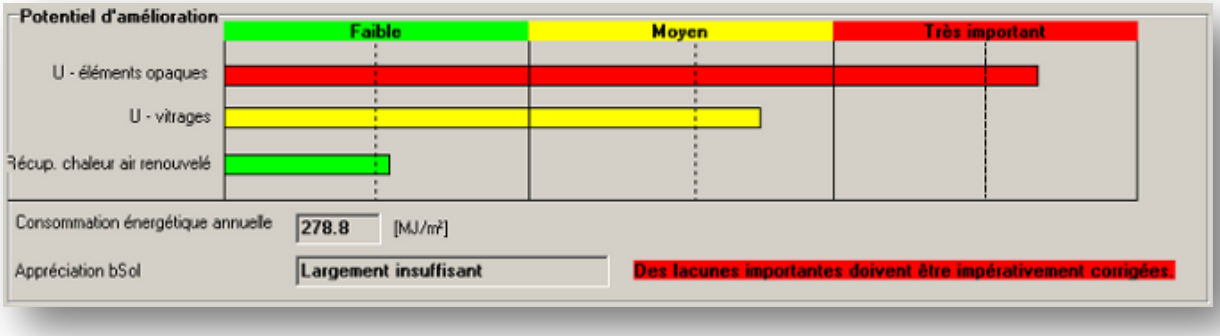


Figure 10: La "boussole" de bSol

Ce logiciel apporte donc une aide décisionnelle non négligeable lors du choix des éléments de construction d'une habitation ou lors de travaux de rénovation.

FONCTIONNEMENT

Le logiciel bSol peut traiter jusqu'à 2 zones (pièces ou ensemble de pièces ayant la même température intérieure) en contact thermique l'une avec l'autre. Pour étudier une zone de bâtiment, bSol effectue de manière horaire, dans le cadre d'un modèle à un nœud, le bilan des gains et des pertes thermiques et détermine ainsi, heure après heure, l'évolution de la température intérieure. Lorsque les conditions de confort l'exigent, la configuration des paramètres variables est modifiée (abaissement de protections solaires mobiles, augmentation de l'aération en vue de rafraîchir); en dernier recours, les installations techniques (chauffage, climatisation) interviennent pour garantir que les conditions du confort intérieur sont remplies.

La somme de toutes les interventions effectuées par les installations techniques sur une certaine période donne l'énergie nette (de chauffage ou de climatisation) qu'il est nécessaire de fournir pour garantir le confort thermique. La valeur maximale de l'énergie fournie par la technique pendant un laps de temps d'une heure sert à déterminer la puissance de pointe nécessaire (de chauffage ou de climatisation).

Le fonctionnement de bSol repose sur l'équation différentielle thermodynamique selon laquelle la puissance thermique amenée au bâtiment équivaut à l'énergie contenue dans les masses internes multipliée par la variation de la température:

$$P = \frac{\Delta Q}{\Delta t} = m \cdot c \cdot \frac{\Delta T}{\Delta t}$$

"Q" étant la chaleur nécessaire pour modifier de "ΔT" la température d'un corps de masse "m".

VALEUR U ET RÉSISTANCE THERMIQUE

La résistance thermique est une des valeurs physiques les plus importantes sur le plan de la protection thermique.

La valeur U ou coefficient de transmission thermique est l'inverse de la résistance thermique. Elle indique la quantité de chaleur qui passe en une seconde à travers 1m² de la surface extérieure d'un élément de construction vers la surface intérieure, avec une différence de température de 1°C ou Kelvin (K) entre les deux surfaces. Elle est indiquée en Watts par mètre carré par Kelvin (W/m²K). Plus cette valeur est petite, meilleure est l'isolation thermique de l'élément de construction et moins il se perd de chaleur.

La valeur U d'un élément de construction dépend de la conductibilité thermique des matériaux utilisés et de leur épaisseur.

Pour calculer la valeur U, il suffit de connaître la résistance thermique et de l'inverser:

$$R_{th} = R_i + \sum_{i=1}^{n} \frac{d_i}{\lambda_i} + R_e$$

$$U = \frac{1}{R_{th}}$$

Rth : résistance thermique totale $\left[\frac{m^2 \cdot K}{W}\right]$

Ri : résistance thermique d'ambiance intérieur, (vaut 0.125 m²K/W)

Re : résistance thermique d'ambiance extérieur (vaut 0.05 m²K/W)

d : épaisseur de la couche "i" $[m]$

λ : conductibilité thermique de la couche "i" $\left[\frac{W}{m \cdot K}\right]$

n : nombre de couches

CONDUCTION THERMIQUE

Dans bSol, un mur (surface) est composé au maximum de cinq matériaux différents (couches). Pour déterminer les températures initiales et finales de chaque couche en ne connaissant que la température initiale et finale du mur entier, Il faut établir un système d'équations à partir de la formule de base de la conduction thermique:

$$P = \frac{dQ}{dt} = \frac{\lambda \cdot S}{d} \cdot (T_1 - T_2)$$

Ceci permettra par la suite de représenter la droite d'échange thermique sur chacune des couches du mur.

P : puissance thermique $[W]$

S : surface $[m^2]$

d : distance (épaisseur) $[m]$

λ : conductibilité thermique $\left[\frac{W}{m \cdot K}\right]$

T_1 : température initiale $[K]$

T_2 : température finale $[K]$

Hypothèses : pas de pertes entre les surfaces ; les surfaces ont la même section « S ».

$$P = \frac{\lambda_1 \cdot S}{d_1} \cdot (T_0 - T_1) \quad \rightarrow \quad T_0 - T_1 = \frac{P \cdot d_1}{\lambda_1 \cdot S}$$

$$P = \frac{\lambda_2 \cdot S}{d_2} \cdot (T_1 - T_2) \quad \rightarrow \quad T_1 - T_2 = \frac{P \cdot d_2}{\lambda_2 \cdot S}$$

$$P = \frac{\lambda_3 \cdot S}{d_3} \cdot (T_2 - T_3) \quad \rightarrow \quad T_2 - T_3 = \frac{P \cdot d_3}{\lambda_3 \cdot S}$$

$$P = \frac{\lambda_4 \cdot S}{d_4} \cdot (T_3 - T_4) \quad \rightarrow \quad T_3 - T_4 = \frac{P \cdot d_4}{\lambda_4 \cdot S}$$

$$P = \frac{\lambda_5 \cdot S}{d_5} \cdot (T_4 - T_E) \quad \rightarrow \quad T_4 - T_E = \frac{P \cdot d_5}{\lambda_5 \cdot S}$$

$$(T_0 - T_1) + (T_1 - T_2) + (T_2 - T_3) + (T_3 - T_4) + (T_4 - T_E) = (T_0 - T_E)$$

$$(T_0 - T_E) = \frac{P \cdot d_1}{\lambda_1 \cdot S} + \frac{P \cdot d_2}{\lambda_2 \cdot S} + \frac{P \cdot d_3}{\lambda_3 \cdot S} + \frac{P \cdot d_4}{\lambda_4 \cdot S} + \frac{P \cdot d_5}{\lambda_5 \cdot S}$$

$$(T_0 - T_E) = \left(\frac{d_1}{\lambda_1} + \frac{d_2}{\lambda_2} + \frac{d_3}{\lambda_3} + \frac{d_4}{\lambda_4} + \frac{d_5}{\lambda_5}\right) \cdot \frac{P}{S}$$

$$P = \frac{(T_0 - T_E) \cdot S}{\frac{d_1}{\lambda_1} + \frac{d_2}{\lambda_2} + \frac{d_3}{\lambda_3} + \frac{d_4}{\lambda_4} + \frac{d_5}{\lambda_5}}$$

$$T_1 = T_0 - \frac{P \cdot d_1}{\lambda_1 \cdot S}$$

$$T_2 = T_1 - \frac{P \cdot d_2}{\lambda_2 \cdot S}$$

Et ainsi de suite…

ANALYSE ET CONCEPTION GÉNÉRALE

STRUCTURE D'UN BÂTIMENT

Dans bSol, un bâtiment est composé de deux zones au maximum, on peut s'imaginer ces zones comme étant la cave non chauffée et la maison qui se trouve au-dessus. Chaque zone peut avoir des masses internes et jusqu'à vingt-quatre faces (face nord, face sud, toiture, etc.) qui sont composées elles-mêmes de surfaces. Une fenêtre est une surface, une porte est une surface, un mur est une surface, chaque face peut contenir ainsi jusqu'à seize surfaces. Finalement, une surface peut être constituée d'un maximum de cinq couches. On imagine très bien le béton, l'isolation et la brique dans le cas d'un mur.

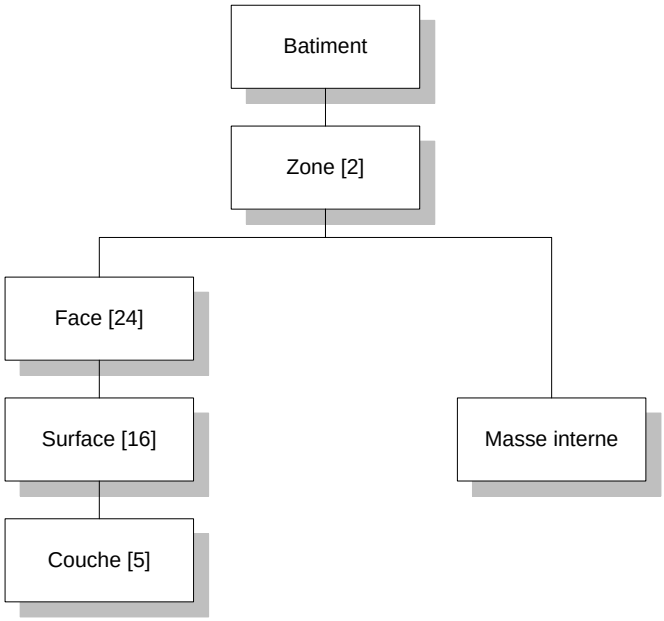


Figure 11: Structure d'un bâtiment

Si l'on regarde plus en détails comment est structurée une surface, on remarque le nombre important de propriétés (variables) qu'elle possède:

```
struct SURFACE   // surface (contact particulier ou ouverture) d'une face ou masse intérieure
{
    char    nom[LEN_NOM_COURT];      // nom de la surface
    int     typeInputU;              // =0 si l'utilisateur entre directement la valeur u sinon ; = 1 si
    float   transmUser;              // transmission entrée par l'utilisateur (si <0 pas d'entrée)
    float   transmConstituant;       // transmission calculée à partir des constituants du mur.
    int     idType;                  // identification du type de surface (envel., fenêtre ou porte)
    float   sTot;                    // superficie totale
    float   longueur;                // longueur de la surface [m]
    float   larg;                    // largeur/hauteur de la surface [m]
    .
    .
    .
    float   reserve[6];
    int     iZoneParent;             // garde l'index de la zone du parent (sur le disque dure)
    int     iFaceParent;             // garde l'index de la face du parent (sur le disque dure)
    int     iSurfaceParent;          // garde l'index de la surface du parent (sur le disque dure)
    struct  SURFACE *surfaceParent;  // adresse de la surface à laquelle on est lié
    int     nCouche;                 // nombre de couches données
    struct  COUCHE couche[NMAX_COUCHE];
};
```

Figure 12: Extrait du code définissant la surface

Il en va de même pour la zone, la face et tous les autres objets nécessaires à définir le bâtiment. Ce ne sont que des structures imbriquées les unes dans les autres.

DESCRIPTION GLOBALE DES PROCESSUS

On se trouve dans la zone *Constituants/Valeur U*.

On a alors la possibilité soit d'entrer manuellement la valeur U, soit de définir nous-mêmes la composition de la surface, soit enfin de lier la surface.

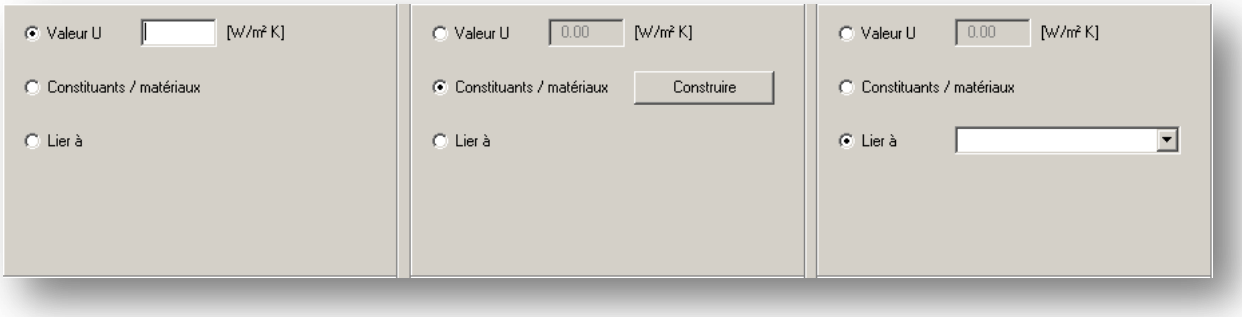


Figure 13: Options de saisie des valeurs U

Si l'on choisit de construire la surface et que celle-ci ne contient encore aucune couche, le bouton affiche "Construire".

Si au contraire, la surface contient déjà des couches, le bouton affiche "Modifier".

En appuyant sur le bouton "Construire", afin de créer un mur, une fenêtre ou bien une porte, l'éditeur de surface correspondant s'ouvre et nous permet de construire la surface avec plusieurs matériaux. Les différents éléments constituant la surface (les couches) sont ensuite enregistrés dans le fichier "bâtiment".

En appuyant sur le bouton "Modifier", la même fenêtre s'ouvre. On peut alors modifier les composants et appliquer le changement.

Lorsque la fenêtre de l'éditeur de surfaces se ferme (ou en appuyant sur un bouton adéquat), la valeur U est envoyée à la zone texte correspondante dans le programme bSol. Cependant, elle ne pourra pas être modifiée sans passer par l'éditeur de surface.

Si des parents existent, on peut, via un bouton devenu visible, appeler la fenêtre permettant de créer de fichiers "bâtiment" avec des valeurs U changeantes.

Depuis la partie *Exploitation* on a la possibilité d'ouvrir une fenêtre similaire à celle des gains internes afin de corriger de manière horaire la consigne de chauffage, de climatisation ou d'aération.

TESTS D'INTÉGRATION

Test	OK	Pas OK
S'assurer de pouvoir importer un cadre "frame" à l'intérieur d'une fenêtre "form".	✓	
Vérifier la rétrocompatibilité des fichiers. Utiliser les bits de réserve le cas échéant.	✓	

LES NOUVELLES FONCTIONNALITÉS DÉVELOPPÉES

LIAISON ENTRE LES SURFACES

Afin d'ajouter les nouvelles fonctions à bSol, la partie *Constituants/Valeur U* va être presque entièrement réécrite. Initialement elle se compose d'un élément de type "Page Control" permettant d'afficher une page différente en fonction du type de surface sélectionné.

Comme on veut avoir un affichage commun à tous les types de surface, on va créer un nouvel objet cadre (Frame). Des instances de ce cadre vont être insérées dans les différentes pages. Quand certains éléments du cadre ne sont pas désirés, il suffit de les rendre invisibles.

Chaque page possédant une instance unique de l'objet cadre, le fait de masquer un élément de l'instance cadre se trouvant dans la page "fenêtre" n'influencera pas le même élément de l'autre instance cadre se trouvant dans la page "mur" ou "porte".

Une nouvelle option "Lier à" va être ajoutée aux deux options déjà existante dans bSol. C'est elle qui permettra de faire des liaisons du type "parent-enfant" entre les surfaces.

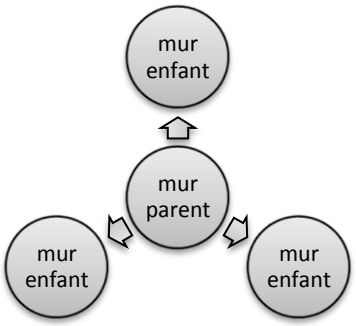


Figure 14: Principe de la liaison

ANALYSE

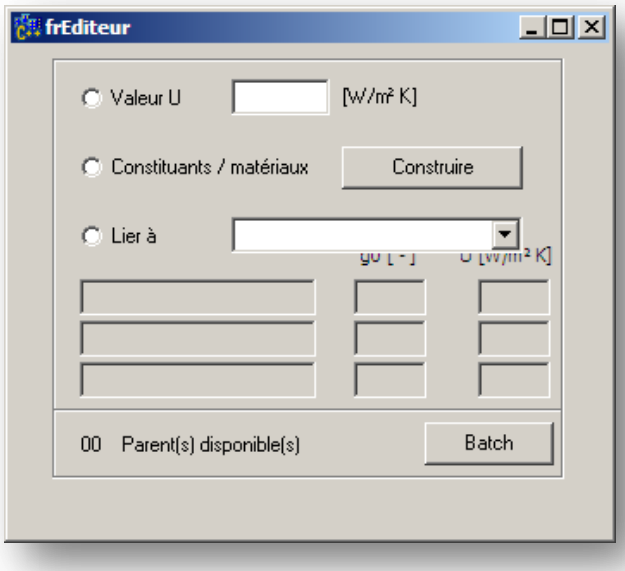


Figure 15:Cadre "frEditeur"

Le cadre "frEditeur" se compose d'un panneau inférieur et d'un panneau supérieur.

Le panneau inférieur contient un élément "Label" pour indiquer le nombre de surfaces parent disponibles et un élément "Bouton" qui appelle la fenêtre de génération de fichiers. Ces éléments ne sont visibles que si des surfaces parent sont disponibles.

Le panneau supérieur contient trois éléments "Bouton Radio" pour le choix de saisie de la valeur U, un élément "Zone Texte" afin d'entrer ou d'afficher la valeur U, un élément "Bouton" qui appelle la fenêtre de l'éditeur de surface, un élément "Liste Déroulante" qui permet de choisir une surface parent, ainsi que plusieurs autres éléments "Zone Texte" et "Label" servant à afficher diverses informations.

CONCEPTION

 En cliquant sur le bouton radio "Valeur U", on décide de saisir la valeur U à la main via la zone texte correspondante. En cliquant sur le bouton radio "Constituants/matériaux", on définit la valeur U par le biais de l'éditeur de surface. En cliquant sur le bouton radio "Lier à", la surface actuelle prend les valeurs de la surface à laquelle on la lie.

Si le bouton radio "Valeur U" est en position sélectionné (sauf mode "fenêtre"):

- La zone texte est visible et active.
- Le bouton construire est invisible.
- La liste déroulante est invisible.
- Les éléments d'information sont invisibles.

Si le bouton radio "Constituants/matériau" est en position sélectionné:

- La zone de texte est visible et inactive.
- Le bouton construire est visible.
- La liste déroulante est invisible.
- Les éléments d'information sont visibles. (sauf mode "mur")

Si le bouton radio "Lier à" est en position sélectionné:

- La zone de texte est visible et inactive. (sauf mode "fenêtre")
- Le bouton construire est invisible.
- La liste déroulante est visible.
- Les éléments d'information sont invisibles.

 La zone texte de la valeur U permet soit de saisir soit d'afficher la valeur. Les zones texte d'informations affichent les coefficients de transmission thermique ou les coefficients de gain solaire des fenêtres.

Si la zone texte de la valeur U est active:

- La valeur U de la surface sélectionnée stockée dans le fichier "bâtiment" est récupérée et affichée.
- On peut saisir la valeur à la main.
- En quittant la zone texte ou en appuyant sur la touche "ENTER", on stocke la valeur de la zone texte dans le fichier "bâtiment".

Si la zone texte est désactivée:

- Elle affiche la valeur U totale des couches de la surface sélectionnée. (sauf mode "fenêtre")
- Elle affiche la valeur U de la surface "parent".

 La liste déroulante permet de sélectionner une surface "parent" à laquelle on veut lier la surface actuelle. La liste est active seulement si la surface actuelle n'est pas elle-même une surface "parent".

Si la liste est activée:

- Par défaut, dans le cas où la surface ne possède pas de "parent", on affiche le premier élément de la liste. Dans le cas contraire, on récupère et on affiche la surface "parent".

Si on déroule la liste:

- Les éléments de la liste sont effacés.
- Toutes les surfaces du bâtiment sont récupérées depuis la base de données et triées afin de n'afficher que les surfaces de même type et n'ayant pas de "parenté".
- Les éléments de la liste sont recréés avec les surfaces pouvant être "parent".

S'il y a un changement:

- L'adresse de la surface parent est stockée dans le fichier "bâtiment".

Le bouton "Construire" permet de choisir les couches de la surface. Pour ce faire, il lance l'éditeur de surface correspondant à la surface sélectionnée.

Si le bouton est actif:

- Contrôle si la surface possède un "parent".
- Affiche le texte "Construire" ou "Modifier" en conséquence.

Si le bouton est cliqué:

- L'adresse de la surface actuellement sélectionnée est placée dans un pointeur global afin qu'on puisse y accéder depuis les éditeurs de surface.
- Lance la fenêtre de l'éditeur de mur. (mode "mur")
- Lance la fenêtre de l'éditeur de fenêtre. (mode "fenêtre")
- Lance la fenêtre de l'éditeur de porte. (mode "porte")

TESTS UNITAIRES

Test	OK	Pas OK
Vérifier les liaisons entre les instances du cadre et la fenêtre bâtiment.	✓	
S'assurer que l'adresse mémoire de la surface "parent" est bien écrite dans le pointeur **SurfaceParent* de la surface "enfant".	✓	

ÉDITEURS DE SURFACE

Les trois éditeurs de surface (mur, fenêtre et porte) sont réalisés sur trois "forms" différentes. Comme la conception de ces éditeurs est sensiblement la même, ils seront tous les trois traités sous ce chapitre.

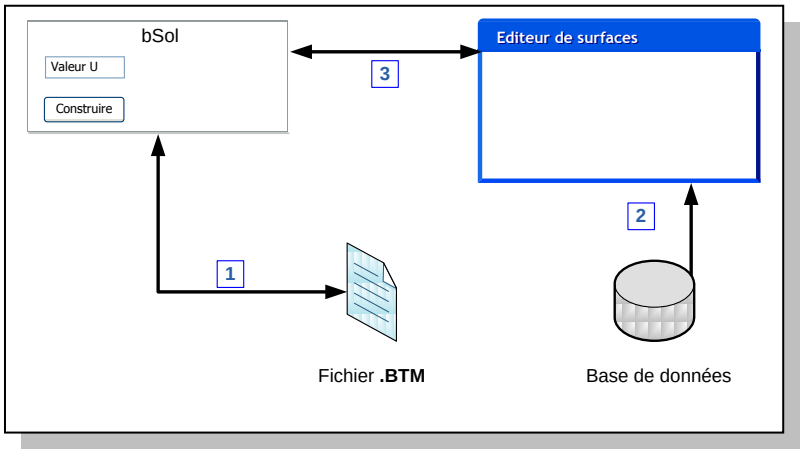


Figure 16: Flux des données

1. bSol ouvre le fichier "bâtiment" et place sont contenu en mémoire.
   bSol prend le contenu placé en mémoire et l'enregistre dans le fichier "bâtiment".
2. L'éditeur de surface lit la base de données des matériaux bSol et récupère les entrées.
3. bSol et l'éditeur de surface partagent le contenu de la mémoire et peuvent le modifier.

Analyse

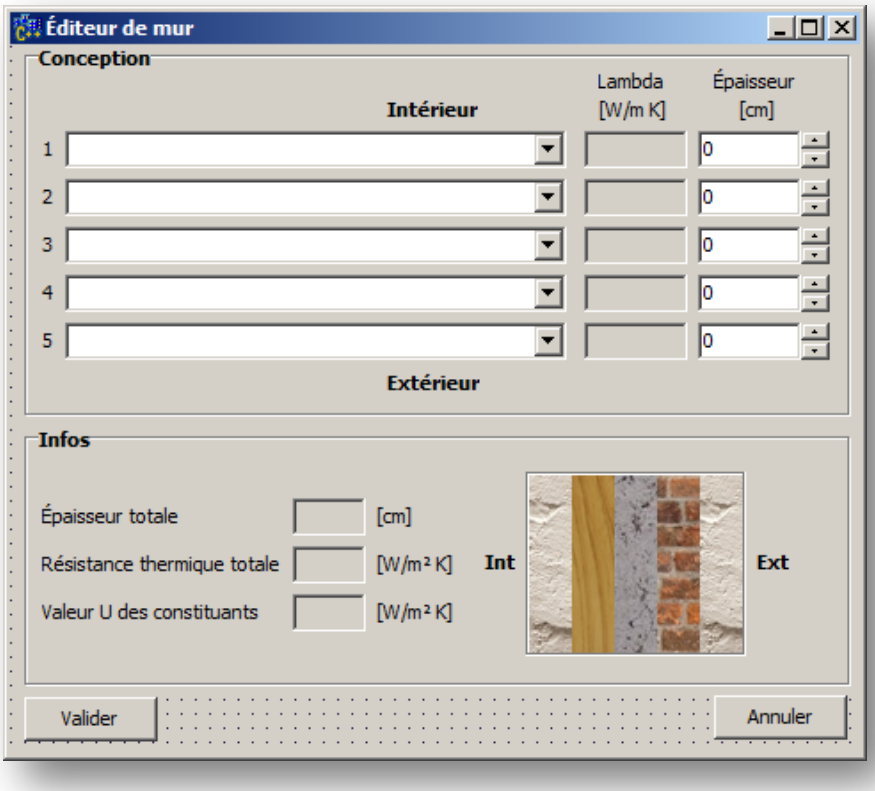


Figure 17: Éditeur de mur

L'éditeur de mur se compose d'un panneau "Conception" et d'un panneau "Infos". Dans le panneau "Conception", on construit notre mur en choisissant cinq matériaux au maximum depuis les listes déroulantes. On peut définir l'épaisseur de chaque couche de matériau dans une zone texte ou la modifier grâce à des boutons fléchés. La conductibilité thermique du matériau sélectionné (λ) est aussi affichée.

Le panneau "Infos" quand à lui, affiche en temps réel la valeur de l'épaisseur totale, la valeur de la résistance thermique totale et la valeur U totale du mur construit. Un graphique nous permet de visualiser une coupe du mur avec ses différentes couches ainsi que la droite d'échange thermique entre l'intérieur et l'extérieur du mur.

Les boutons "Annuler" et "Valider" permettent respectivement de fermer la fenêtre de l'éditeur de mur et d'envoyer les valeurs vers bSol.

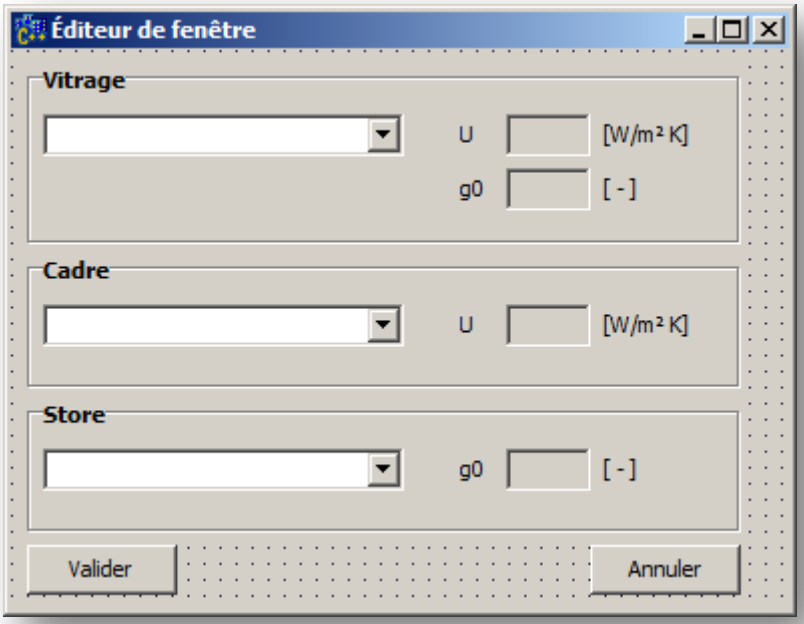


Figure 18: Éditeur de fenêtre

L'éditeur de fenêtre se compose de trois panneaux (Vitrage, Cadre et Store), chaque panneau contient une ou deux zones texte affichant la valeur U et le coefficient de gain solaire (g0), ainsi qu'une liste déroulante afin de choisir le matériel à utiliser.

Les boutons "Annuler" et "Valider" permettent respectivement de fermer la fenêtre de l'éditeur de fenêtre et d'envoyer les valeurs vers bSol.

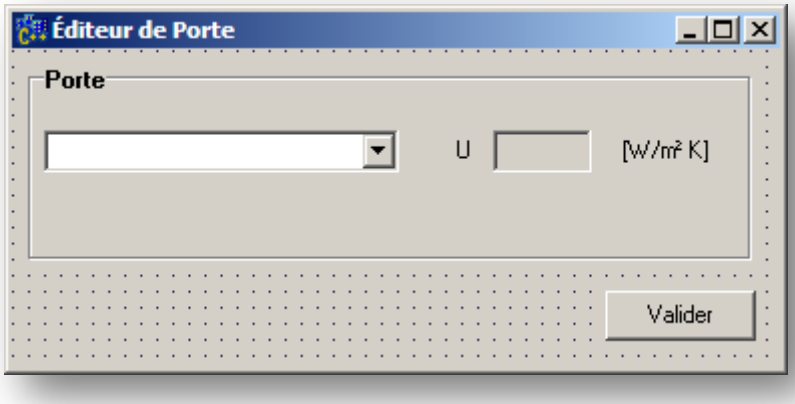


Figure 19: Éditeur de porte

L'éditeur de porte possède un unique panneau dans lequel le choix de la porte se fait par l'intermédiaire d'une liste déroulante et l'affichage de la valeur U par une zone texte.

Le bouton "Valider" permet d'envoyer les valeurs vers bSol.

CONCEPTION

 Les listes déroulantes des matériaux.

Si on déroule les listes:

- Les éléments de la liste sont effacés.
- Les matériaux de la base de données sont récupérés et triés.
- Les éléments de la liste sont recréés avec les matériaux correspondants.

S'il y a un changement:

- La conductibilité du matériau est récupérée.
- L'épaisseur du matériau est récupérée.
- La valeur U et le coefficient de gain solaire sont récupérés et affichés dans les zones texte correspondantes. (mode "fenêtre")
- La résistance thermique et la valeur U sont calculées est affichées si elles sont différentes de zéro. (mode "mur")
- Les images du graphique sont actualisées est mises à la bonne échelle.
- La droite d'échange thermique est calculée et affichée.

 Les zones texte de l'épaisseur.

Si la valeur change:

- L'épaisseur totale est calculée et affichée.
- La résistance thermique et la valeur U sont calculées est affichées si elles sont différentes de zéro.
- Les images du graphique sont actualisées est mises à la bonne échelle.
- La droite d'échange thermique est calculée et affichée.

 Le bouton "Valider"

Si le bouton est cliqué:

- La valeur U totale est stockée dans le fichier "bâtiment".
- Les propriétés des matériaux et les épaisseurs sont stockées dans le fichier "bâtiment".

PROGRAMMATION

Tout d'abord il y a le calcul de la valeur de résistance thermique. Pour la trouver, et ainsi trouver aussi le coefficient de transmission thermique qui n'est autre que son inverse, nous utilisons la formule suivante (voir chap. *bSol*):

$$R_{th} = R_i + \sum_{i=1}^{n} \frac{d_i}{\lambda_i} + R_e$$

En code, cela donne:

```
Rt = fmBatiment->RThermIntExt;                  // Rt contient la somme de Ri et Re
resist = 0;

for(int i = 0; i < 5; i++)
{
    int k= cbl[i]->ItemIndex;                   // index de l'élément de la liste déroulante
    if(k<0) continue;                           // évite le cas où k = -1.

    if(opaque[k].U > 0.00001)
    {
        Rt += 1/opaque[k].U;                    // calcul la résistance Th avec Ri et Re
        resist += 1/opaque[k].U;                // calcul la résistance Th sans Ri et Re
    }
}
if(Rt != fmBatiment->RThermIntExt)              // n'afficher pas les valeurs quand les épaisseurs sont 0
{
    edRTot->Text = FormatFloat("0.00",Rt);
    edValeurU->Text = FormatFloat("0.00",1/Rt);
}else{
    edRTot->Clear();
    edValeurU->Clear();
}
```

Figure 20: Extrait du code

Ensuite, pour dessiner la droite d'échange thermique sur chaque couche, il faut connaître les températures sur les deux faces de la couche. L'utilisation de la formule suivante nous permet donc de trouver la température sur la face d'arrivée en connaissant la température de la face de départ (voir chap. *bSol*) :

$$T_1 = T_0 - \frac{P \cdot d_1}{\lambda_1 \cdot S}$$

Cependant, la puissance thermique et la surface sont encore inconnues. Pour remédier à cela, la température intérieur T_O et la température extérieur T_E du mur sont définies comme des valeurs constantes. La surface aussi sera égale à 1. Cela n'a aucune répercussion sur les calculs de bSol car ces valeurs ne sont utilisées que pour dessiner de la droite.

Les valeurs des épaisseurs et des lambdas sont récupérées dans les zones texte.

$$P = \frac{(T_0 - T_E) \cdot S}{\frac{d_1}{\lambda_1} + \frac{d_2}{\lambda_2} + \frac{d_3}{\lambda_3} + \frac{d_4}{\lambda_4} + \frac{d_5}{\lambda_5}}$$

En code, cela donne :

```
if(ep[0]+ep[1]+ep[2]+ep[3]+ep[4]!= 0)                    // vérifie que les épaisseurs ne soient pas égales à 0
{
    puissanceTherm = ((Tchaud-Tfroid)*S) / (img[0]->Width/lambda[0] + img[1]->Width/lambda[1] +
                img[2]->Width/lambda[2] + img[3]->Width/lambda[3] + img[4]->Width/lambda[4]);
}else{
    puissanceTherm = 0;                                  // le cas où toutes les épaisseurs sont zéro
}
for(int i = 0; i < 5; i++)                               // dessine les droites
{
    if(i == 0)
    Tinit[i] = Tchaud;
    else
    Tinit[i] = Tfinal[i-1];

    if(i == 4)
    Tfinal[i] = Tfroid;
    else
    Tfinal[i] = Tinit[i] - ((puissanceTherm * img[i]->Width) / (lambda[i] * S));

    img[i]->Picture->LoadFromFile(bitmap[i]);            // Recharge l'image afin de supprimer le Canvas.
    if(i == 0)
    img[i]->Canvas->MoveTo(0, 0);
    else
    img[i]->Canvas->MoveTo(0, old);

    if(i<4)
    img[i]->Canvas->LineTo(img[i]->Width, 100-Tfinal[i]);
    else
    img[i]->Canvas->LineTo(122 - img[i]->Left, 100-Tfinal[i]);

    old = 100-Tfinal[i];
}
```

Figure 21: Extrait du code

TESTS UNITAIRES

Test	OK	Pas OK
Contrôler que la base de données soit correctement lue.	✓	
Vérifier l'exactitude du calcul de la résistance thermique et de la valeur U en comparant le résultat obtenu avec la version non modifiée de bSol.	✓	
Vérifier que la valeur U, g0 et les couches sont bien écrites dans la mémoire en allant lire les valeurs des adresses "transmConstituant" et "couche".	✓	
Vérifier que la droite d'échange thermique soit cohérente.	✓	

LE GÉNÉRATEUR DE FICHIERS

Ce module n'est disponible que s'il existe des surfaces "parent" dans le bâtiment.

ANALYSE

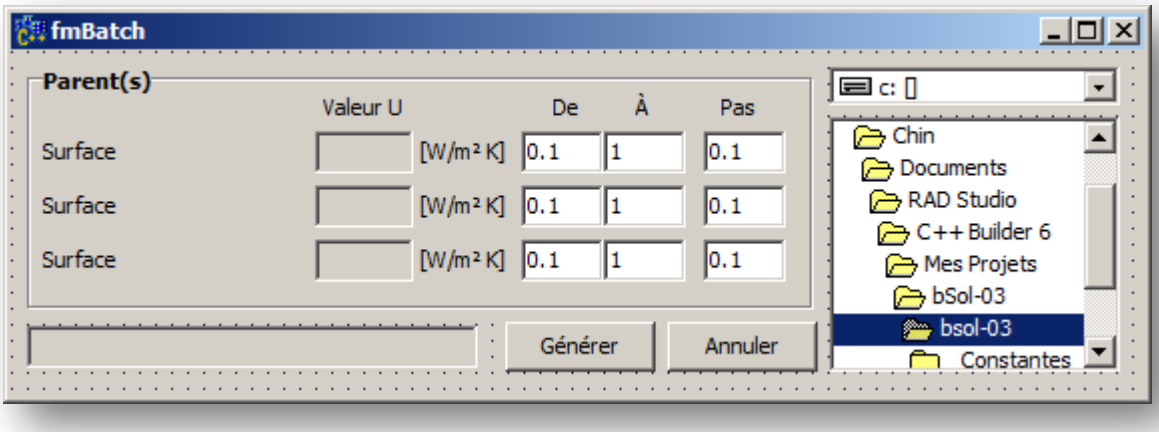


Figure 22: Générateur de fichiers

Le générateur se compose d'un panneau *Parent(s)* à l'intérieur duquel son disposés des éléments *Label* qui affichent le nom des différentes surfaces "parent" (3 au maximum). Les zones texte permettent de définir un intervalle dans lequel on veut faire varier la valeur U de la surface "parent" ainsi que le pas de variation.

On a aussi la possibilité de choisir l'emplacement du dossier dans lequel on veut sauvegarder les fichiers générés.

CONCEPTION

La fenêtre "fmBatch"

Lorsque la fenêtre apparaît:

- Une copie des surfaces "parent" est d'abord stockée dans la mémoire temporairement.

Lorsque la fenêtre est fermée:

- La ou les surfaces stockées précédemment sont rappelées afin qu'on se retrouve dans l'état initial.
- L'espace mémoire est libéré.

Le bouton "Générer"

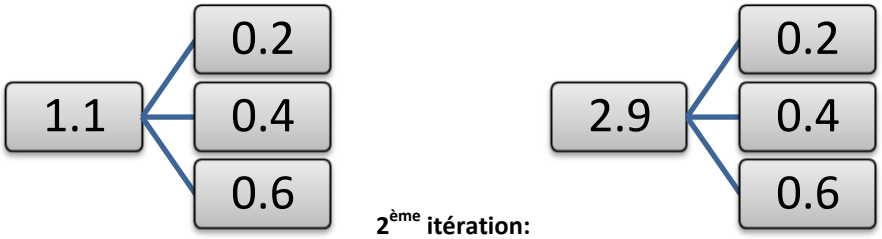
Si le bouton est cliqué:

- Les valeurs U sont incrémentées en fonction des valeurs des zones texte.
- Les fichiers "bâtiment" sont créés au fur et à mesure.

PROGRAMMATION

Afin de générer des fichiers dont les valeurs U changent, et de traiter tous les cas possibles de façon dynamique, l'utilisation d'une fonction récursive s'impose.

Examinons par exemple le cas où l'on veut qu'une fenêtre prenne les valeurs U {1.1; 2.9} et qu'un mur prenne les valeurs U {0.2; 0.4; 0.6}. Au final, on obtiendra six fichiers car pour chaque cas de fenêtre on doit varier de trois fois la valeur U du mur:



1ère itération: **2ème itération:**

Figure 23: principe de fonctionnement

Jusque là, pas de problème, on connaît le nombre de surfaces à calculer. Une fonction simple peut le faire.

Imaginons maintenant que l'on veuille supprimer la fenêtre, ou encore ajouter une porte. Les choses se compliquent car la fonction ne doit plus traiter seulement deux surfaces mais s'adapter à un nombre de surfaces non connu d'avance.

Il faut donc créer une fonction qui "s'appelle" elle-même autant de fois qu'il y a de surfaces:

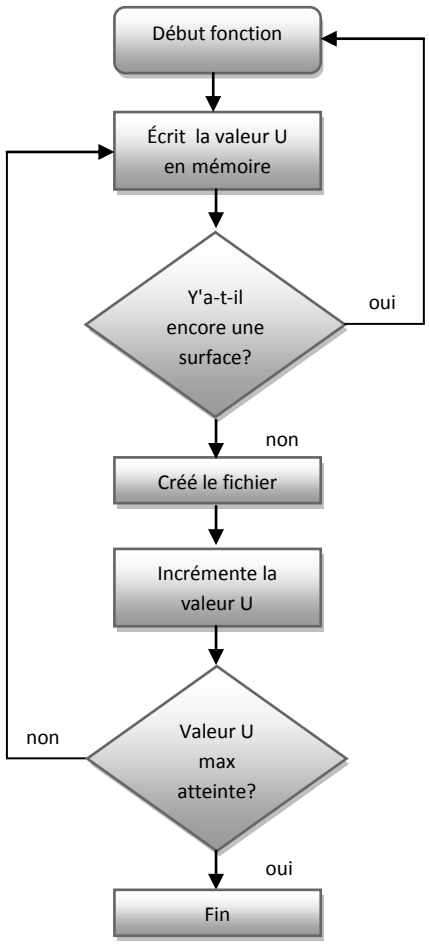


Figure 24: Structogramme simplifié de la fonction récursive

En code, cela donne :

```
void __fastcall TfmBatch::GenererFichiers(TObject *Sender)
{
    if(de[fmBatch->idParent]->Text.ToDouble() < a[fmBatch->idParent]->Text.ToDouble())
    {
        for(float i = (float)de[fmBatch->idParent]->Text.ToDouble();
        i <= (float)a[fmBatch->idParent]->Text.ToDouble();
        i += (float)pas[fmBatch->idParent]->Text.ToDouble())          // intervalle des valeurs U
        {
            ecrisValeurU(fmBatch->idParent, i);                       // écriture en mémoire

            if(fmBatch->idParent == fmBatiment->nbrParents - 1)       // dernière surface ?
             creeFichier();                                           // génère le fichier

             if(fmBatiment->nbrParents > 1)
            {
                if(fmBatch->idParent < fmBatiment->nbrParents - 1)    // s'il y a encore des surfaces
                {
                    fmBatch->idParent++;
                    GenererFichiers(Sender);                          // la fonction s'appelle même
                }
            }
         }
        if(fmBatiment->nbrParents > 1)
        fmBatch->idParent--;
    }
}
```

Figure 25: Extrait du code

TESTS UNITAIRES

Test	OK	Pas OK
Vérifier que les fichiers son créés sur le disque dur, à l'endroit souhaité.	✓	
Vérifier que les valeurs écrites dans les fichiers sont exactes en ouvrant ces derniers avec bSol.	✓	
Vérifier que le nombre correct de fichiers est généré.		✗

LES CONSIGNES DANS LE MODE EXPERT

ANALYSE

Dans la version originale de bSol, seules les charges internes (personnes et appareils) peuvent être modifiées de manière graphique en utilisant une méthode multiplicative:

Exemple: Si la consigne est de 3.0 $\left[\frac{W}{m^2}\right]$ et que pendant les heures du jour et les jours de la semaine on l'utilise à 80%, mais que pendant les semaines de l'année on la garde à 100%, on se retrouve avec une nouvelle consigne de 1.9 $\left[\frac{W}{m^2}\right]$ soit $3.0 \cdot 0.8 \cdot 0.8 \cdot 1 = 1.9$

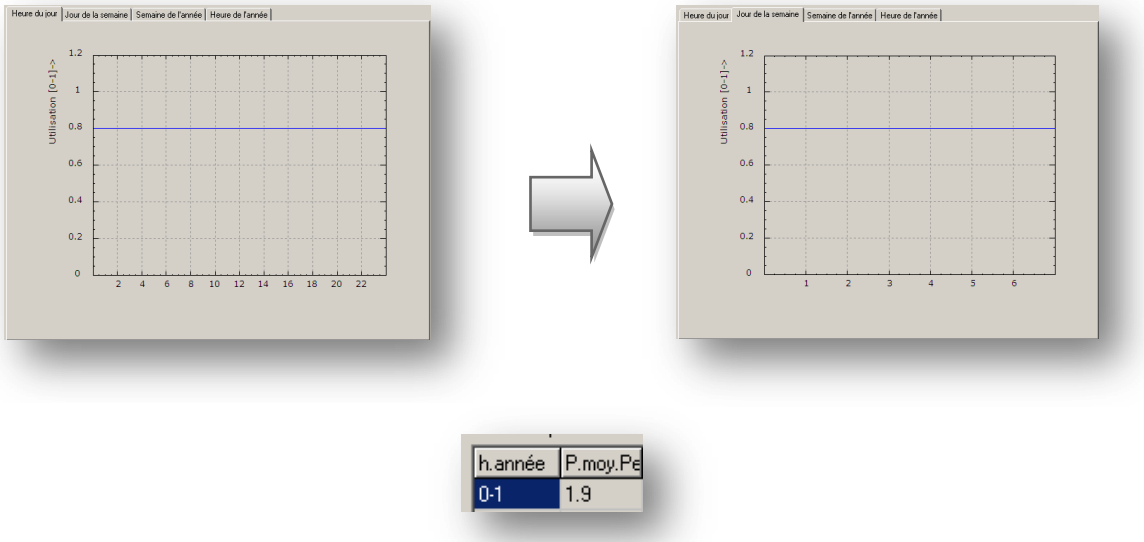


Figure 26: Principe de la consigne multiplicative

Dorénavant, on souhaite que toutes les consignes soient modifiables graphiquement mais uniquement dans le mode expert et avec une méthode un peu plus simple.

CONCEPTION

La méthode choisie est la suivante: tout d'abord, on commence par définir l'exploitation pendant les semaines de l'année avec les unités réelles de la consigne. Ensuite on ajuste cette courbe durant les jours de la semaine puis les heures du jour en modifiant le taux d'utilisation de la consigne. Prenons pour exemple le cas du chauffage d'une école:

- En premier lieu on place la consigne de chauffage (22°C) et la valeur minimale (4°C).
- Ensuite on dessine la courbe de l'exploitation, 22°C pendant les 26 premières semaines de l'année puis on descend à la valeur minimale jusqu'à la 33ème semaine ce qui correspond à éteindre le chauffage durant les vacances d'été. On remonte à 22°C jusqu'aux vacances de Noël puis on éteint à nouveau le chauffage.
- Notons que lorsqu'une courbe de niveau supérieur est modifiée (valeur différente de 22°C) elle n'est plus touchée par la modification des courbes de niveau inférieur (semaines→jours→heures).
- On passe cette fois aux jours de la semaine, la courbe représente maintenant le taux de la valeur de consigne. Celle-ci est à 100% (22°C) du lundi au vendredi et passe à 0% (4°C) le weekend.
- Rappelons ici que les vacances d'été et de Noël n'ont pas été influencées.
- Enfin on procède de la même manière pour les heures du jour, 100% de 7h00 à 17h00 puis 0% de 17h00 à 7h00.

On se retrouve donc avec une approche plus intuitive de l'exploitation d'un bâtiment. Et ce processus est le même pour la climatisation, les gains internes, l'aération et la gestion des stores.

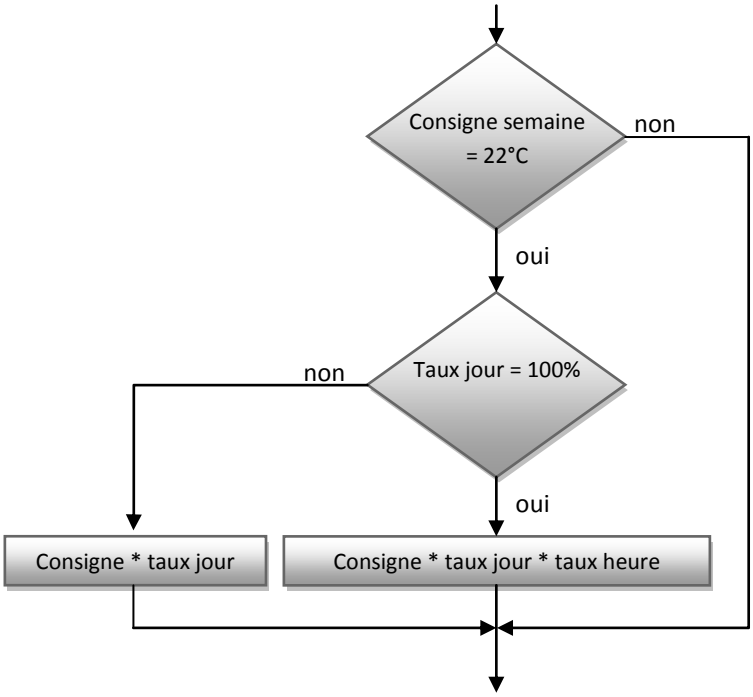


Figure 27: Structogramme de la méthode de saisie de la consigne

Toutes les valeurs sont en suites placées dans un tableau de 8760 heures.

Cette méthode ayant quand-même des limites, il a été décidé de pouvoir modifier manuellement et individuellement chaque valeur du tableau grâce à une interface utilisateur du type Excel. Afin de ne pas utiliser de logiciel tierce partie dans bSol, le choix du composant "*StringGrind*" s'est naturellement posé.

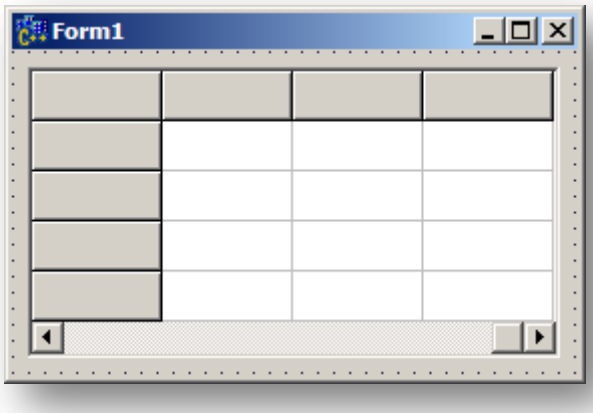


Figure 28: Composant "StringGrind"

Le composant "*StringGrind*" se présente comme un tableau à ce détail près qu'il ne gère pas le copier-coller de plusieurs cellules avec Excel. Un algorithme a donc été développé pour lire et placer des données dans le presse-papier.

Lorsqu'on copie des données depuis un programme comme Excel, celles-ci sont formatées de la manière suivante dans le presse-papier :

donnée ***/t*** *donnée* ***/r/n*** *donnée* ***/t*** *donnée*

Les cellules sont séparées par des tabulations "/t" et les lignes par des retours à la ligne "/r/n". L'algorithme analyse donc le texte se trouvant dans le presse papier et à chaque "/t", il passe à la cellule suivante du composant "*StringGrind*" de même qu'à chaque "/r/n" il passe à la ligne suivante.

```
void __fastcall TextToCells(TStringGrid& AGrid, const TGridCoord& AFirstCell, const AnsiString AText)
{
   TGridCoord cell = AFirstCell;

   int word_start = 1;
   const int text_len = AText.Length();
   for (int index = 1; index < text_len; ++index)
   {
      char current_char = AText[index];
      // si c'est une nouvelle colonne
      if (current_char == '\t')
      {
         AGrid.Cells[cell.X++][cell.Y] = AText.SubString(word_start, index - word_start);
         // saute le caractère '\t'
         word_start = index + 1;
      }
      // si c'est une nouvelle ligne
      else if (current_char == '\r')
      {
         AGrid.Cells[cell.X][cell.Y++] = AText.SubString( word_start, index - word_start);
         cell.X = AFirstCell.X;
         // saute les caractère '\r' et '\n'
         word_start = index + 2;
      }
   }
}
```

Figure 29: Extrait du code

TESTS UNITAIRES

Test	OK	Pas OK
Contrôler que les graphiques soient correctement interprétés.	✓	
Vérifier le bon fonctionnement du code en contrôlant les valeurs se trouvant dans le tableau.	✓	

LES GRAPHIQUES D'ÉDITION DE LA CONSIGNE

ANALYSE

Afin d'éditer les consignes des gains internes, bSol affichait une fenêtre à partir de laquelle étaient appelés les graphiques correspondant à la consigne choisie. Chaque graphique possède son identifiant:

Identifiants des graphiques	Heures/année	Semaines/année	Jours/semaine	Heures/jour
Personnes	200	220	240	260
Appareils	210	230	250	270

Les méthodes servant à afficher les graphiques possèdent deux paramètres: "**zone**" utilisé pour savoir dans quelle zone on se trouve et "**mode**" pour identifier les personnes ou les appareils.

CONCEPTION

Afin de pouvoir modifier les autres valeurs de consigne de l'exploitation, on a besoin d'autres graphiques et d'autre identifiants. Pour garder une certaine cohérence, les identifiants on été définis comme suit:

Identifiants des graphiques	Heures/année	Semaines/année	Jours/semaine	Heures/jour
Personnes	200	220	240	260
Appareils	210	230	250	270
Chauffage	300	320	340	360
Climatisation	310	330	350	370
Aération	400	420	440	460
Stores	410	430	450	470

Et comme il était souhaité de garder une fenêtre unique pour l'affichage des graphiques, un nouveau paramètre ("**position**") est ajouté aux méthodes d'appel de ces derniers. Ainsi, avec les trois paramètres, on peut clairement savoir à quel type de consigne on a affaire.

Paramètre "zone"	Paramètre "position"	Paramètre "mode"	Consigne
0	0	0	Personnes
		1	Appareils
	1	0	Chauffage
		1	Climatisation
	2	0	Aération
		1	Stores

Test	OK	Pas OK
S'assurer que le passage du mode basique au mode expert se fasse correctement.	✓	
Vérifier que les liaisons de calcul sont faites en simulant un bâtiment dont la consigne de chauffage est dynamique en mode expert. Puis en simulant le même bâtiment avec une consigne constante mais en ajustant les heures de début du jour et de la nuit en mode basique.	✓	

La fenêtre Exploitation

Analyse

La fenêtre "exploitation" de bSol se présente à la façon d'un rectangle allongé en hauteur. Elle est divisée en plusieurs parties à savoir:

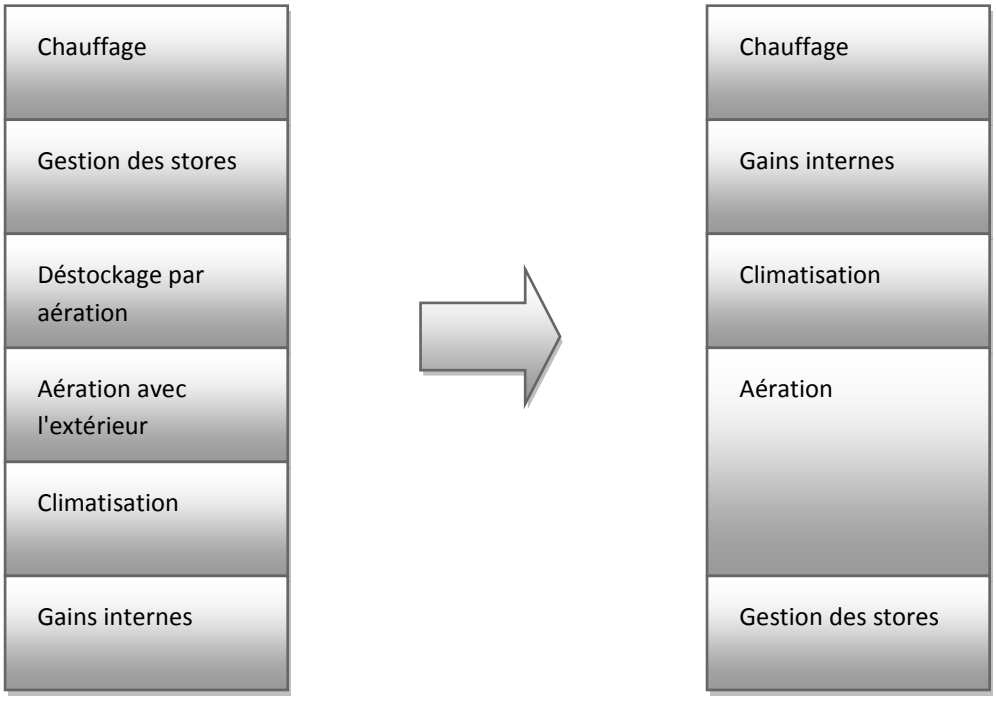


Figure 30: Représentation de la fenêtre "exploitation"

Elle a été repensée de façon à regrouper les apports de chaleur dans la partie du haut et les déperditions dans la partie du bas. De plus, chaque partie dispose maintenant d'un bouton dans le mode expert permettant la modification de la consigne sur un graphique.

Figure 31: Passage en mode expert

OMBRAGE DES FENÊTRES

Actuellement, il existe sur bSol une propriété "Facteur d'ombrage" permettant de définir le pourcentage de la fenêtre se trouvant à l'ombre. Ce facteur est statique, c'est-à-dire qu'il ne varie jamais.

En réalité, lorsqu'une fenêtre est exposée à une zone d'ombre, cela vient généralement de la projection de l'ombre d'un obstacle se trouvant entre le soleil et la fenêtre. Ceci implique donc que cette zone varie constamment en fonction de la position du soleil.

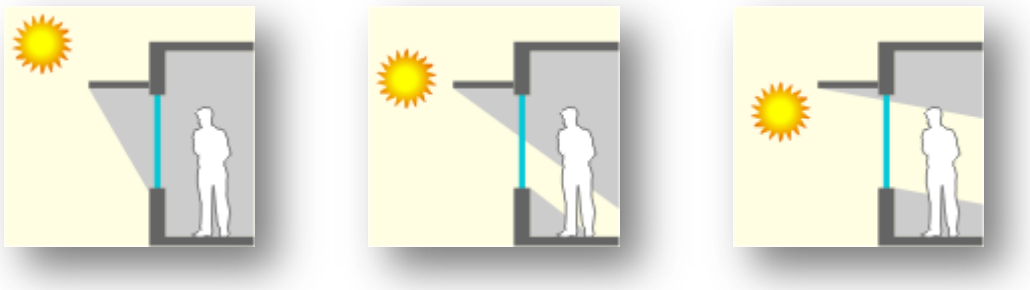


Figure 32: Influence du soleil sur les surfaces ouvertes

Il a été donc décidé d'ajouter une fonctionnalité permettant de simuler le facteur d'ombrage de la fenêtre en fonction de la position du soleil lorsque celle-ci possède un auvent ou des écrans latéraux.

ÉCRAN HORIZONTAL

Dans le cas d'un auvent et d'une fenêtre rectangulaire, il existe des courbes (indicateur d'occultation) permettant d'étudier les protections solaires. On peut imaginer ces courbes comme un gabarit de lignes d'ombre pour la verticale et pour l'horizontale.

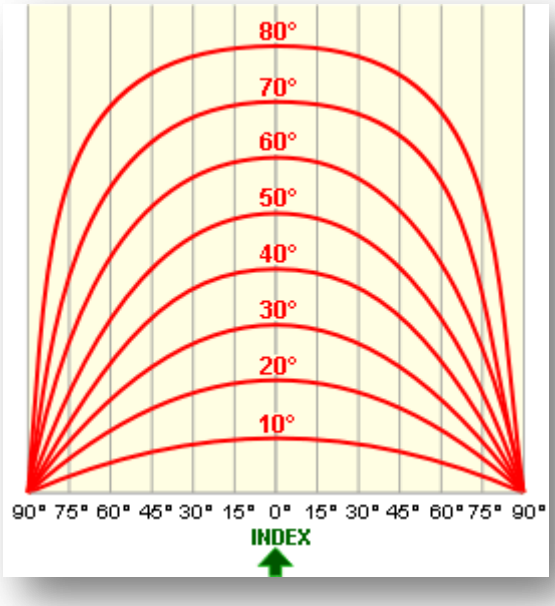


Figure 33: Indicateur d'occultation

Comme on connaît la géométrie de la fenêtre, on peut déterminer une plage sur l'indicateur d'occultation correspondant aux différents pourcentages d'ombre de la fenêtre.

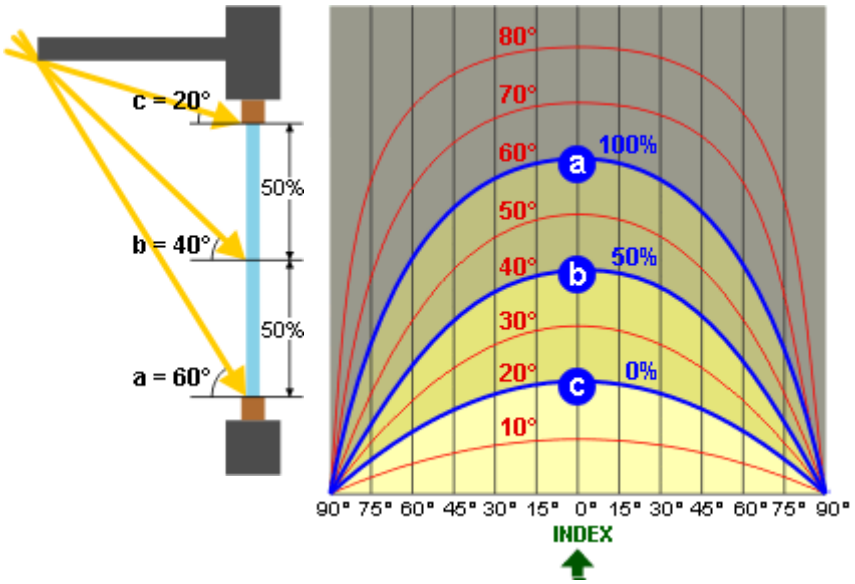


Figure 34: Zone d'ombre pour un auvent

Par exemple, si le soleil se trouve à une hauteur de 40° et possède une orientation nulle par rapport au plan verticale perpendiculaire à la fenêtre, on constate que le 50% de celle-ci se trouve à l'ombre. On se trouve aussi à 50% d'ombrage lorsque le soleil se trouve à une hauteur d'environ 30° et est orienté à environ 50°.

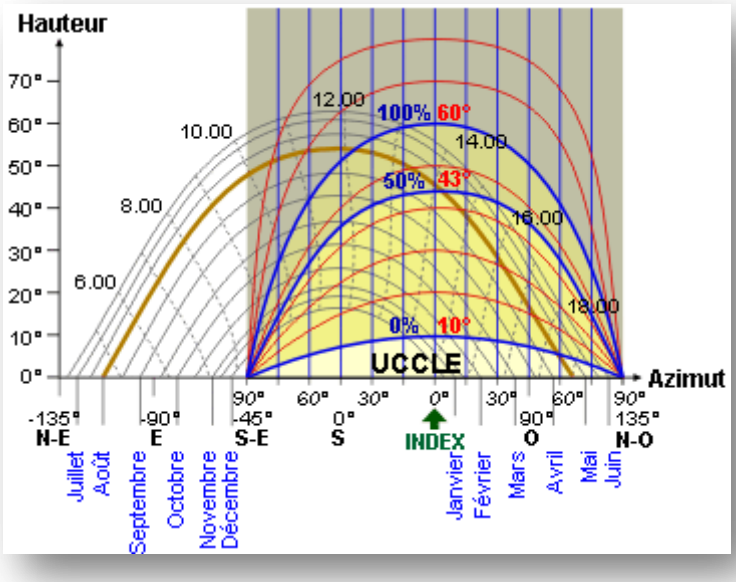


Figure 35: Diagramme solaire superposé à l'indicateur d'occultation

En superposant cet indicateur au diagramme solaire du site étudié, on détermine graphiquement les instants pendant lesquels la fenêtre reçoit ou ne reçoit pas de rayonnement solaire. Afin de pouvoir arriver à ce même résultat par calcul, il faut connaître les équations de ces lignes d'ombre. Ne trouvant aucune formule ou calcul à ce sujet, il a fallu déterminer une équation approchant au mieux l'indicateur d'occultation.

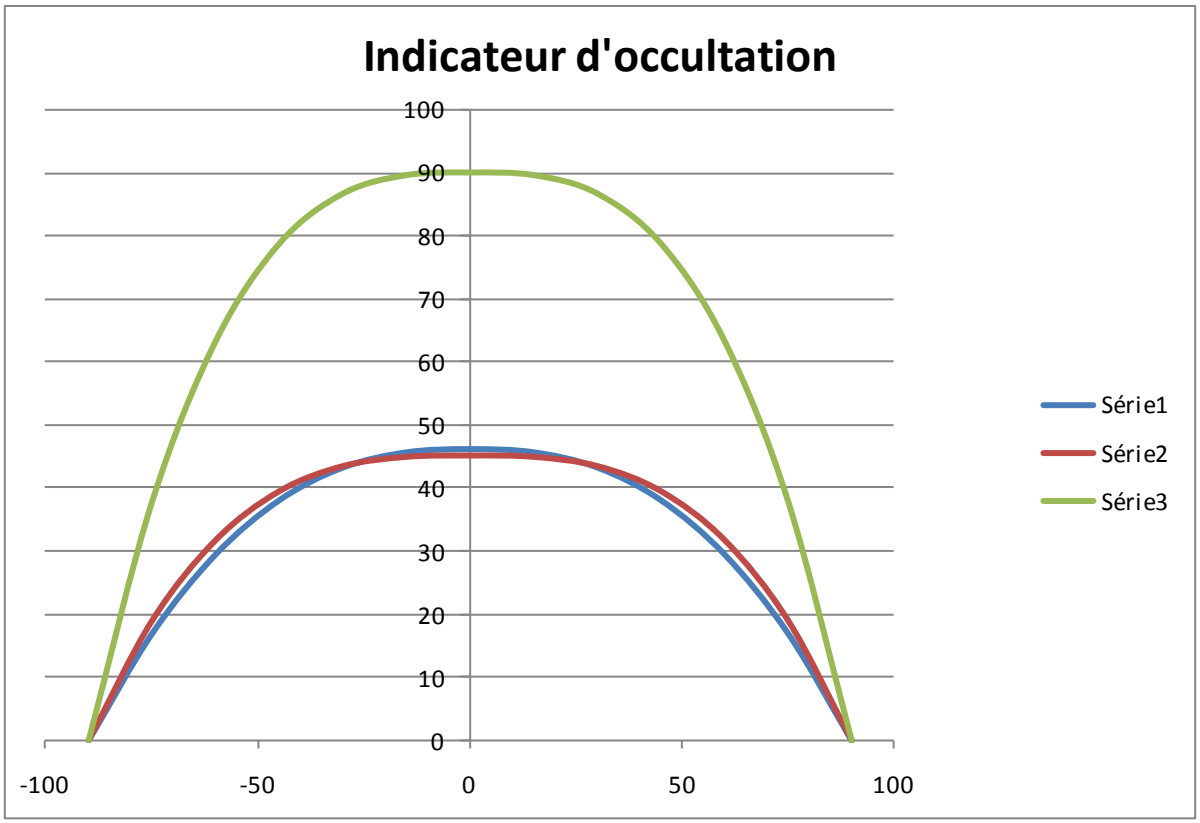


Figure 36: Approximation des courbes d'occultation

Une fois l'équation trouvée, on peut maintenant déterminer le facteur d'ombrage de la fenêtre en fonction de la hauteur et de l'azimut solaire.

ÉCRANS VERTICAUX

Dans le cas d'écrans verticaux sur les côtés de la fenêtre, on procède de la même manière en utilisant cette fois les lignes verticales de l'indicateur pour déterminer un profil d'ombrage à partir de la géométrie de la fenêtre.

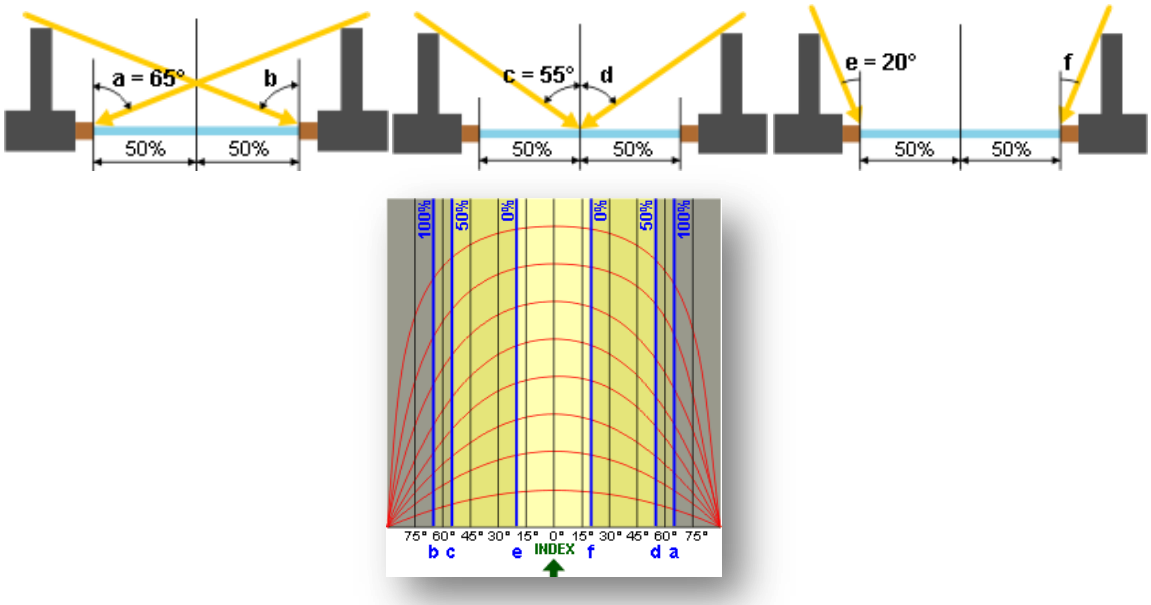


Figure 37: Zone d'ombre pour des écrans latéraux

Ainsi, en combinant le profil du auvent et celui des écrans latéraux, on obtient le profil d'ombre d'un ensemble pare-soleil comportant des parties horizontales et verticales.

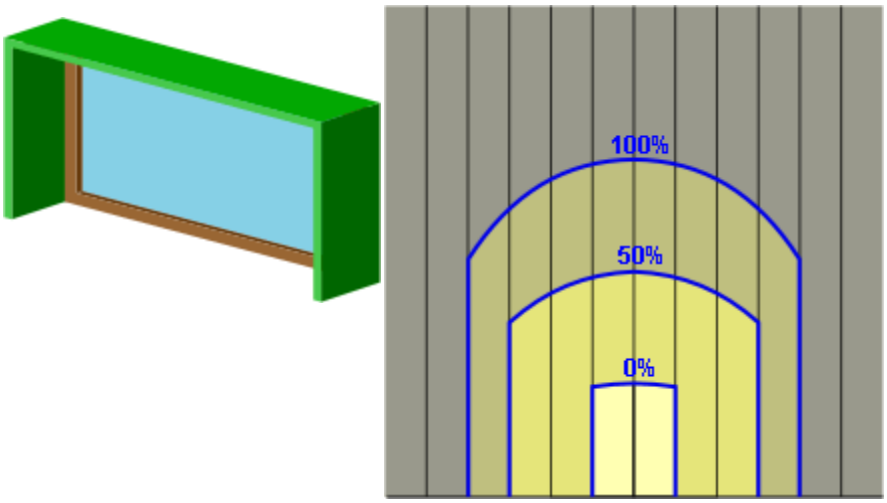


Figure 38: Profil d'ombre pour un système de pare-soleil

À l'heure actuelle, le module pour le calcul du facteur d'ombrage est intégré à bSol mais n'est pas encore fonctionnel car les liaisons avec la partie de calculs n'est pas finalisée.

TESTS

TEST DU BÂTIMENT

Il s'agit ici de vérifier que le logiciel bSol fonctionne correctement et que les calculs sont exacts. On va donc procéder à un test du bâtiment. Tout d'abord, un bâtiment est créé, ensuite les différents éléments sont ajouté et définis en utilisant les nouvelles fonctionnalités implémentées (éditeurs de surface, liaisons, etc.). Après avoir lancé le calcul, voilà le résultat obtenu:

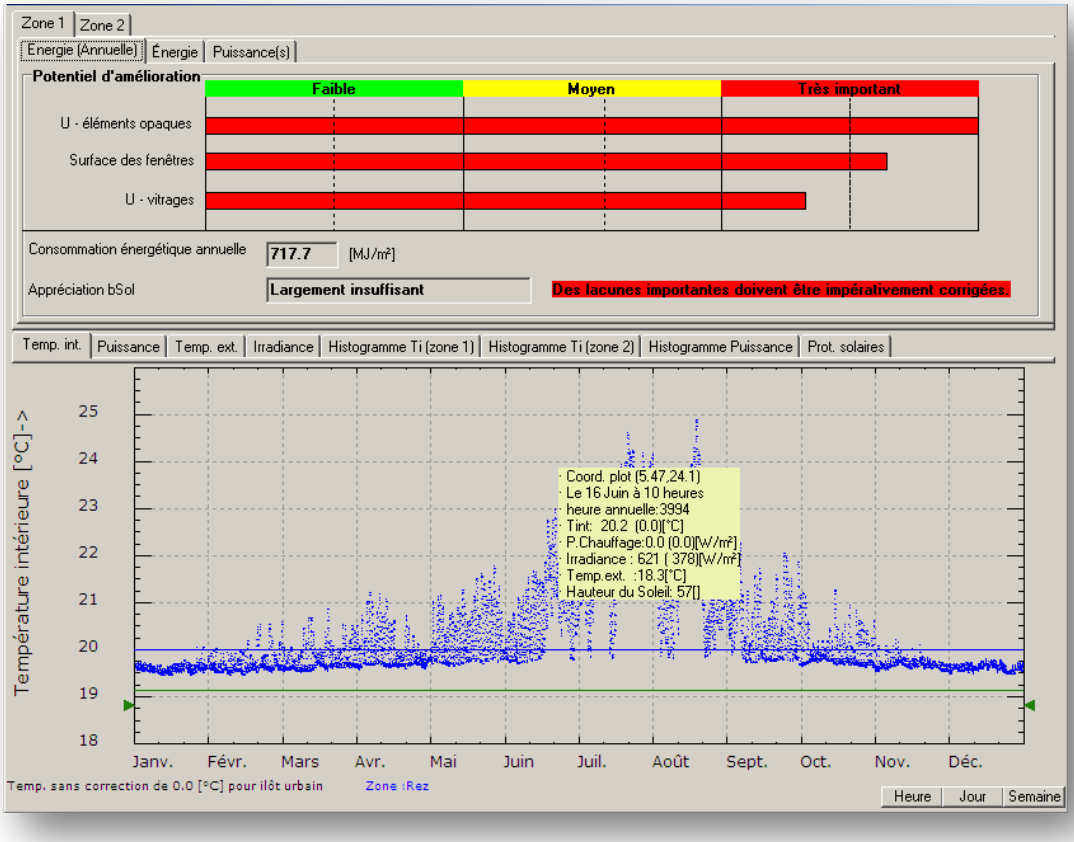


Figure 39: Résultat du calcul avec la nouvelle version de bSol

Ensuite le même bâtiment, avec la même météo, la même exploitation est créé dans l'ancienne version du logiciel bSol. Évidemment il n'y a plus l'option de liaison, on va donc construire de façon identique les surfaces qui étaient liées lors du calcul précédent. Le résultat obtenu est le suivant:

Figure 40: Résultat du calcul avec l'ancienne version de bSol

Après avoir réalisé le test plusieurs fois avec des bâtiments différents, on peut en conclure que le test du bâtiment a été réalisé avec succès.

TEST DE L'EXPLOITATION

Comme bSol possède désormais un mode basique et un mode expert, il est nécessaire de tester ce dernier afin d'en assurer le bon fonctionnement.

CONSIGNE DE CHAUFFAGE

En simulant une consigne de chauffage constante de 22°C pour un bâtiment donné en mode expert, on obtient une consommation annuelle de 440.6 $\left[\frac{MJ}{m^2}\right]$.

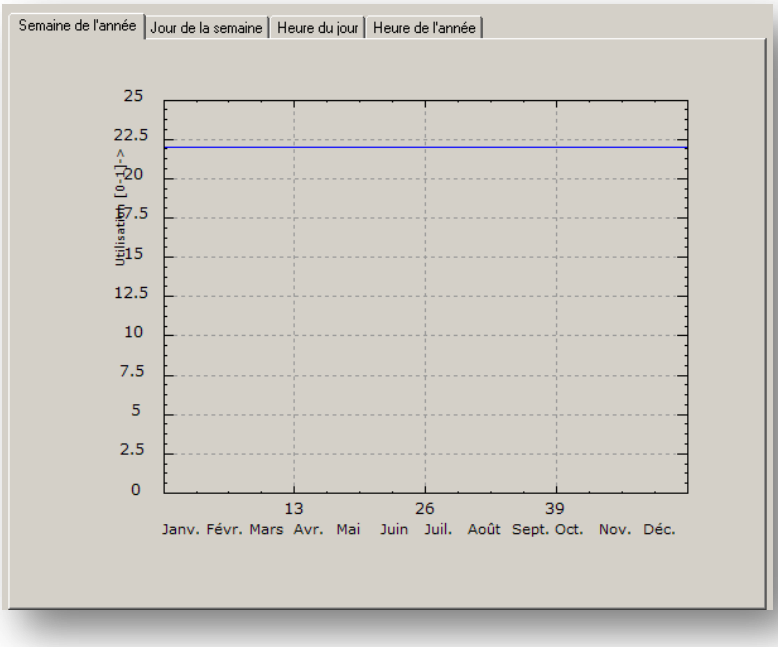


Figure 41: Consigne de chauffage en mode expert

En faisant la simulation avec l'ancien bSol et en définissant une température de jour et une température de nuit de 22°C, on obtient exactement le même résultat.

En simulant maintenant une consigne de chauffage variable dans le mode expert (22°C de 0h00 à 12h00 puis 13°C de 12h00 à minuit), la une consommation annuelle de 385.9 $\left[\frac{MJ}{m^2}\right]$.

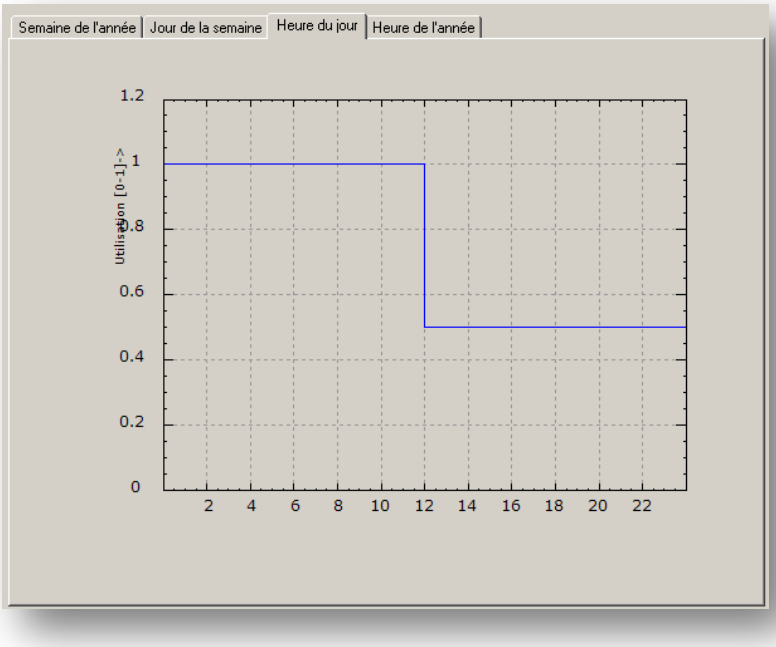


Figure 42: Consigne de chauffage en mode expert

En effectuant la simulation cette fois-ci avec l'ancien bSol pour une température de jour à 22°C et une température de nuit à 13°C, puis en mettant le début du jour à 0h00 et le début de la nuit à 12h00, le résultat est aussi le même.

CONSIGNE DE CLIMATISATION

Pour une consigne de climatisation de 25°C modifier sur le graphique expert à 26°C, la simulation donne une consommation annuelle de 1.9 $\left[\frac{MJ}{m^2}\right]$.

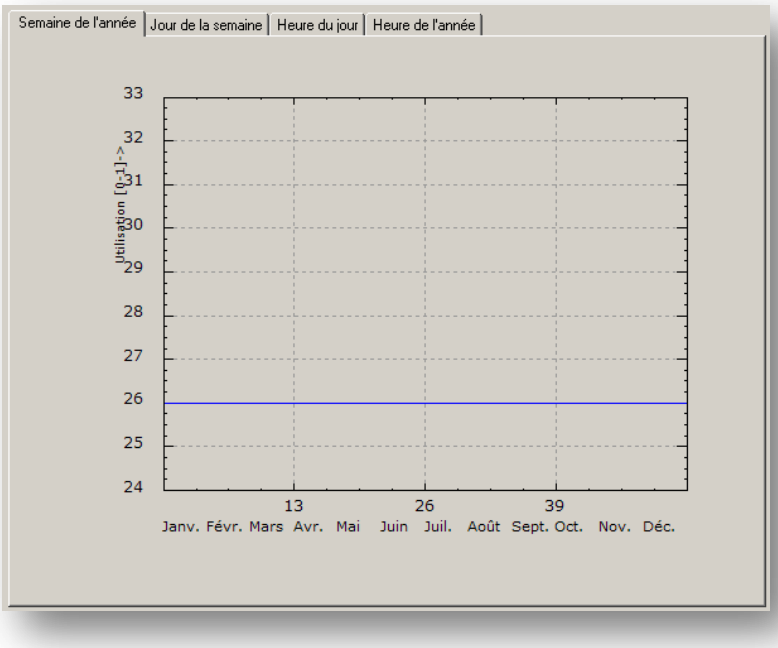


Figure 43: Consigne de climatisation en mode expert

Si on simule une consigne fixe de 26°C avec l'ancienne version de bSol, le résultat est identique.

AUTRES CONSIGNES

Des tests similaires on été réalisés avec les consignes des gains internes et de l'aération en ayant toujours un résultat identique à celui de la version non modifiée de bSol. Seuls les tests de la gestion des stores n'ont pu être effectués car cette fonctionnalité n'est pas encore finalisée à l'heure actuelle.

PROBLÈMES NON ATTENDUS

Malgré le bon fonctionnement du logiciel, certaines "imperfections" subsistent dans le code de programmation. Cependant, une bonne partie de ces problèmes a pu être contournée.

INTERFACE BSOL

Lorsqu'on stocke une surface "parent" dans le fichier "bâtiment", on ne stocke en fait que son adresse mémoire. Hors, si on lance bSol sur un ordinateur différent ou après l'avoir fermé et exécuté d'autres applications, les adresses mémoire ne sont plus forcément les mêmes car elles sont gérées par le système d'exploitation. Donc si l'on veut retrouver l'adresse mémoire de la surface "parent" et que celle-ci n'existe plus, il y a là un problème.

☑ Solution trouvée : afin de retrouver les bons "parents", on ajoute trois variables à la structure "SURFACE" et on y enregistre les numéraux d'index (récupérés depuis l'objet *TreeView*) de la zone, de la face et de la surface "parent".

ÉDITEUR DE MUR

Les boutons fléchés  servant à incrémenter et à décrémenter l'épaisseur fonctionnent très bien du moment que la valeur n'est pas entrée à la main. Si tel est le cas, la décrémentation ne se fait que jusqu'à la valeur entrée mais ne descend pas en dessous.

☑ Solution trouvée : Les boutons fléchés possèdent un paramètre nommé "position" qui est de type *int* (entier) et qui va de 0 à l'infini. Cette valeur s'incrémente lorsque la flèche "up" est cliqué et se décrémente lorsque la flèche "down" est cliquée. Ceci implique que lorsque une valeur est entrée à la main dans le champ de saisie, par exemple la valeur 3, le paramètre "position" des boutons fléchés reste à 0 ou à sa valeur actuelle. Il est donc possible d'augmenter la valeur mais si "position" est à 0, on ne peut pas aller plus bas. Lorsqu'une valeur est entrée, on actualise le paramètre position en lui donnant deux fois la valeur entrée. Ainsi, il est possible de faire varier l'entrée par pas de 0.5. En effet, entre 0 et 3 il y a 6 positions (0.5, 1, 1.5, 2, 2.5, 3).

DROITE D'ÉCHANGE THERMIQUE

Lorsque l'on dessine la droite d'échange thermique, il s'agit de donner le point de départ avec la fonction *MoveTo(x,y)* et le point d'arrivée avec la fonction *LineTo(x,y)*, une ligne est ensuite créée entre ces deux points. La ligne est dessinée sur le bitmap représentant la couche du mur. Or, lorsqu'il faut redessiner la droite afin de l'actualiser, il est impossible d'effacer la ligne précédemment dessinée. Au final, cela donne un gribouillis de lignes indéchiffrable.

☑ Solution trouvée : à chaque fois que l'on actualise la droite, le bitmap est d'abord rechargé.

GÉNÉRATEUR DE FICHIERS

Les variables contenant les valeurs U que l'on souhaite modifier sont du type *float* (décimale à virgule flottante), ceci implique que lorsque l'on place par exemple la valeur 0.6 dans cette variable que l'on nommera "x", parfois, celle-ci contient en réalité la valeur 0.6000001. Donc, quand on test la condition *if(x <= 0.6)*, celle-ci n'est pas remplie alors qu'elle le devrait être. Il arrive ainsi que tous les fichiers ne soient pas créés.

☒ Pas de solution trouvée.

VARIATION DE G0 POUR LE GÉNÉRATEUR DE FICHIERS

Ici, la variation de la valeur U d'une fenêtre est réalisée en faisant varier la valeur U de son vitrage. Cependant chaque vitrage possède un coefficient de gain solaire qui lui est propre. Le problème se pose lorsque l'on veut faire passer la valeur U, par exemple, de 1.1 à 1.5 par pas de 0.1. Aucun vitrage se trouvant dans la base de données ne dispose de valeurs U égales à 1.2 ou 1.4, il est donc impossible de récupérer les coefficients de gain solaire pour ces cas.

☑ Solution trouvée : on évalue tous les vitrages contenus dans la base de données et on détermine une équation liant les valeurs U aux coefficients de gain solaire par l'intermédiaire d'une courbe de tendance :

$$U = 0.1796 \cdot \log(g0) + 0.5754$$

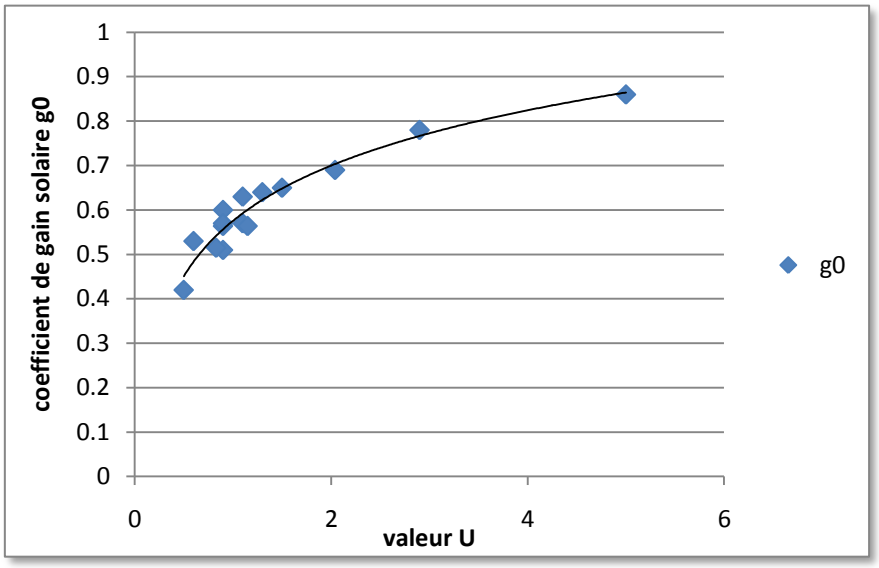


Figure 44: Courbe de tendance pour les coefficients de gain solaire

LES TABLEAUX DYNAMIQUES

On désire placer des valeurs dans un tableau, le nombre de valeurs n'est pas constant, il faut donc que le tableau change de taille. Ceci est réaliser en allouant de la mémoire au tableau à l'aide des méthodes *malloc()* et *realloc()*. Cependant, il arrive que lorsque l'on alloue à nouveau de la mémoire, les valeurs contenues dans le tableau changent subitement.

☑ Solution trouvée : on déclare un tableau de taille finie en laissant assez de réserve pour des nouvelles entrées. Bien entendu, cette solution n'est pas optimale.

ZONE CLIQUABLE DES GRAPHIQUES D'ÉDITION DE LA CONSIGNE

Lorsqu'on édite une courbe de consigne, par exemple les semaines de l'année, et que l'on relâche le bouton de la souris en dehors du graphique; imaginons qu'on se trouve à la semaine 54 (qui n'existe pas), et bien les

valeurs des 54 semaines sont placées dans un tableau. Hors, le tableau en question a une taille de 52 semaines, les 2 autres semaines restantes sont donc placées n'importe où dans la mémoire.

☑ Solution trouvée : les valeurs de dépassement ne sont maintenant plus prise en compte grâce à un contrôle conditionnel.

LES SEMAINES DE L'ANNÉE

L'année comporte 8760 heures, 365 jours et 52.14 semaines soit 52 semaines plus 24 heures. Sur les graphiques de bSol affichant les heures de l'année, les dernières 24 heures du mois de décembre sont manquantes car les tableaux contenant les valeurs des semaines sont déclarés avec une taille de 52. Les 24 heures manquantes font en fait partie de la 53ème semaine.

☑ Solution trouvée : en déclarant à nouveau les tableaux des semaines avec une taille de 53, le problème est contourné.

AMÉLIORATIONS

AMÉLIORATIONS RÉALISÉES

APERÇU DES COUCHES DANS L'ÉDITEUR DE MUR

Pour assurer la liaison entre les images les matériaux et la base de données sans modifier celle-ci on va utiliser la place non utilisée du *char* contenant le nom du matériau. Pour entrer le nom du matériau, on dispose d'un tableau de *char* d'une taille de 80 caractères, on supposera que les 80 caractères ne seront jamais utilisés en totalité. Ainsi, les deux dernier caractères vont être utilisée pour l'identifiant de l'image associée, il y aura donc possibilité d'identifier 100 images car avec les 2 caractères, on peut aller de 0 à 99.

```
70 6C 6F 74 73 20 64 65 20 63 69 6D 65 6E 74 20 ; plots de ciment
70 65 72 66 6F 72 E9 73 00 CD CD CD CD CD CD CD ; perforés.ÍÍÍÍÍÍÍ
CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD ; ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD ; ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
CD CD CD CD CD CD CD CD CD CD CD CD CD CD 30 33 ; ÍÍÍÍÍÍÍÍÍÍÍÍÍÍ03
5A 65 6D 65 6E 74 73 74 65 69 6E 65 20 67 65 6C ; Zementsteine gel
6F 63 68 74 00 00 00 CD CD CD CD CD CD CD CD CD ; ocht...ÍÍÍÍÍÍÍÍÍ
CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD ; ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD ; ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD ; ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
70 65 72 66 6F 72 61 74 65 64 20 63 6F 6E 63 72 ; perforated concr
65 74 65 20 62 6C 6F 63 6B 73 00 CD CD CD CD CD ; ete blocks.ÍÍÍÍÍ
CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD ; ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD ; ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD ; ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
```

Figure 45: Espace occupé par la variable "char" de 80 caractères

Afin de pouvoir lire ce chiffre qui se compose de 2 caractères, on va d'abord les concaténer dans un tableau de 2 caractères avec la fonction *sprintf*, puis convertir ce nouveau *char* en *int* avec la fonction *atoi*. Le tour est joué, il ne reste plus qu'a associer le chiffre à une image.

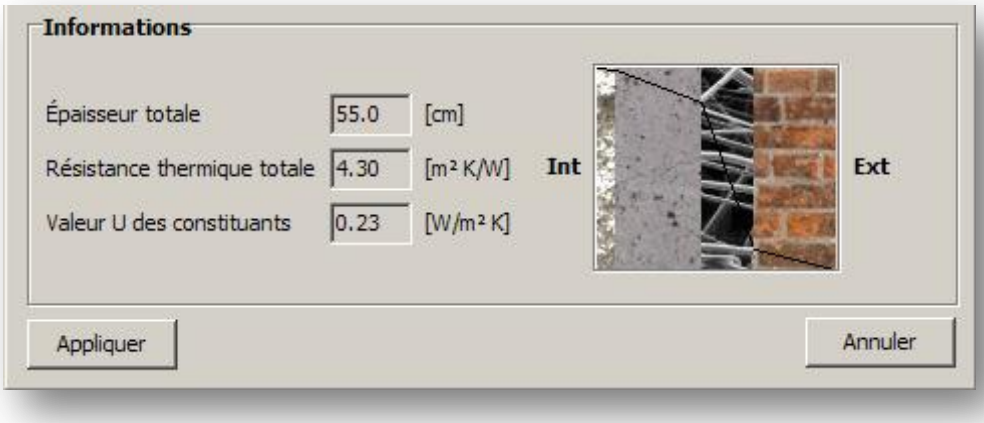


Figure 46: Aperçu des couches

GÉNÉRATEUR DE FICHIERS

Avec la nouvelle fonction permettant de générer les fichiers "bâtiment", il devient plus aisé de traiter et analyser des variations de grandes quantités de données. Pour l'illustrer, voilà ce que l'on peut facilement obtenir en combinant bSol avec un logiciel pouvant créer des plots:

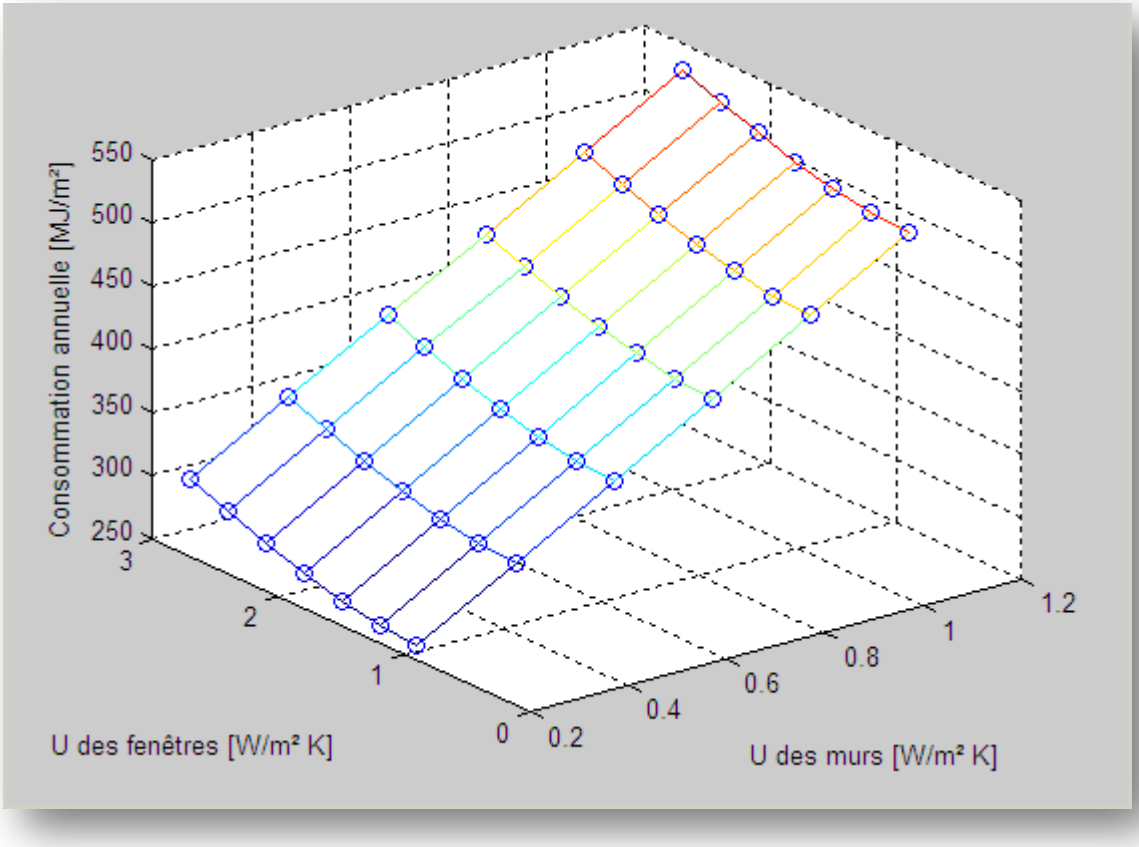


Figure 47: Analyse d'un bâtiment pour différents cas de valeurs U

CE QU'IL RESTE À FAIRE

Par manque de temps, certaines modifications plus ou moins importants ont été négligées au profit de tâches plus urgentes mettant quelques fois en péril le bon fonctionnement du logiciel. Voici certains points inachevés:

- La traduction linguistique des nouvelles fonctionnalités implémentées.
- Finaliser la liaison du module d'ombrage dynamique avec les calculs de bSol.
- Finaliser l'implémentation de la consigne des stores en modes expert.
- Supprimer la 2ème zone.
- Utiliser une nouvelle méthode d'enregistrement des fichiers.
- Exporter un rapport conforme à la norme SIA.

AIDE ET RÉFÉRENCES

LES LIENS

- http://www.bsol.ch/
- http://www.functionx.com/bcb/index.htm
- http://www.cplusplus.com/

LES PERSONNES

- Morand Gilbert-André (professeur responsable)
- Seppey Pierre-André

CONCLUSION

Pour conclure, je peux affirmer maintenant que j'éprouve un sentiment de satisfaction pour avoir mené à terme ce travail de diplôme qui, un peu déroutant au début par rapport au manque de rapprochement avec l'orientation *Power and Control*, s'est avéré très instructif et même passionnant au fil des jours.

Mes notions en programmation et compréhension de la gestion de la mémoire, alors faibles et d'un niveau très basique, n'en sont que plus solides aujourd'hui. L'enthousiasme ainsi que le suivi fréquent et régulier assuré par mon professeur responsable ont énormément contribué à ce résultat.

D'une manière générale, les principaux points du cahier des charges ont été respectés et remplis. Mais suite à plusieurs changements d'idée et à la volonté de modifier le logiciel bSol de façon plus fondamentale, certaines fonctionnalités n'ont pas pu être finalisées à temps. Bien que certaines complications imprévues soient apparues en cours de route, les tests et validations se sont tous déroulés avec succès.

ANNEXES

Un Cdrom contenant le code de programmation.